

# Competitive content-based video copy detection using global descriptors

Juan Manuel Barrios · Benjamin Bustos

© Springer Science+Business Media, LLC 2011

**Abstract** Content-Based Video Copy Detection (CBVCD) consists of detecting whether or not a video document is a copy of some known original and to retrieve the original video. CBVCD systems rely on two different tasks: Feature Extraction task, that calculates many representative descriptors for a video sequence, and Similarity Search task, that is the algorithm for finding videos in an indexed collection that match a query video. This work details a CBVCD approach based on a combination of global descriptors, an automatic weighting algorithm, a pivot-based index structure, an approximate similarity search, and a voting algorithm for copy localization. This approach is analyzed using MUSCLE-VCD-2007 corpus, and it was tested at the TRECVID 2010 evaluation together with other state-of-the-art CBVCD systems. The results show that this approach enables global descriptors to achieve competitive results and even outperforms systems based on combination of local descriptors and audio information. This approach has a potential of achieving even higher effectiveness due to its seamless ability of combining descriptors from different sources at the similarity search level.

**Keywords** Video copy detection · Metric spaces · Automatic weighting · Approximate search · Multimedia information retrieval

---

J. M. Barrios (✉) · B. Bustos  
PRISMA Research Group, Department of Computer Science, University of Chile,  
Av. Blanco Encalada 2120 3er Piso, 8370459, Santiago, Chile  
e-mail: jbarrios@dcc.uchile.cl

B. Bustos  
e-mail: bebustos@dcc.uchile.cl

## 1 Introduction

Multimedia Information Retrieval (MIR) aims at searching and retrieving multimedia documents from large collections. MIR systems can be classified in two approaches: Text-Based, when the search is performed using the textual information associated with a document (tags, titles, and metadata in general), and Content-Based (CBMIR), when the search is based on the multimedia content itself (colors, edges, textures, etc.). Content-Based methods can improve the effectiveness of retrieval even when textual information is present [20]. However, the lack of coincidence between the information that can be extracted from the multimedia document and the interpretation that a user gives to the same data is known as the Semantic Gap [30].

As a part of CBMIR, Content-Based Video Copy Detection (CBVCD) consists of detecting whether or not a video document is a copy of some known original and to retrieve the original video. CBVCD systems perform the detection using only the audio and visual content of videos, ignoring any metadata associated with videos. In general, the techniques used in CBVCD are similar to those used in CBMIR, however the main difference is while CBMIR tries to bridge the semantic gap, CBVCD aims to retrieve a multimedia document even when some transformations have been applied to them.

In this paper, we present an approach for CBVCD called P-VCD, which is based on global descriptors and metric spaces. P-VCD was developed by the PRISMA Research Group at the University of Chile for its participation in TRECVID 2010 [2]. A summarized overview of P-VCD can be found in [3].

The main contributions of this work are: a detailed review of the proposed approach, a novel automatic weight selection algorithm for combination of descriptors, a deep analysis of the system parameters and capabilities. This work is the product of careful analyses and experiments to justify that the global descriptors can indeed be competitive. Algorithms 1–5 may have been cited from [3], but we included them and adapted them to the formalism of this paper in order to make clearer the discussion in the experimental section.

In summary, this work shows a successful application of the metric space approach to the CBVCD problem. It proves that simple global descriptors can achieve high effectiveness and even outperform systems based on local descriptor and/or audio information.

The rest of the paper is structured as follows: Section 2 reviews the current work for CBVCD. Section 3 shows our approach in detail. Section 4 presents the experimental evaluation and analysis of results. Section 5 concludes the paper giving some final thoughts and recommendations about global descriptors.

## 2 Background and related work

In this section, we review the metric space approach for similarity searches and its application for MIR. Then, we review the foundations of CBVCD with the most commonly used techniques for feature extraction and similarity search.

## 2.1 Metric spaces

Let  $\mathcal{D}$  be a set of objects (called the domain), and let  $d : \mathcal{D} \times \mathcal{D} \rightarrow \mathbb{R}$  be a function (called the distance), a metric space [32]  $\mathcal{M}$  is defined by the pair  $(\mathcal{D}, d)$ , where  $d$  satisfies the metric properties:

reflexivity	$\forall x, y \in \mathcal{D}, d(x, x) = 0$
positiveness	$\forall x, y \in \mathcal{D}, x \neq y \Rightarrow d(x, y) > 0$
symmetry	$\forall x, y \in \mathcal{D}, d(x, y) = d(y, x)$
triangle inequality	$\forall x, y, z \in \mathcal{D}, d(x, z) \leq d(x, y) + d(y, z)$

Given a collection  $\mathcal{R} \subseteq \mathcal{D}$ , and a query object  $q \in \mathcal{D}$ , the range search returns all the objects in  $\mathcal{R}$  that are closer than a distance threshold  $\epsilon$  to  $q$ . The nearest neighbor search ( $k$ -NN) returns the  $k$  closest objects to  $q$  in  $\mathcal{R}$ , and the NN+range search returns the intersection between a  $k$ -NN search and a range search.

In the case of Image Retrieval systems,  $d$  is usually a dissimilarity function between images that compares their global descriptors. Let  $\mathcal{I}$  be the set of images, let  $g : \mathcal{I} \rightarrow \mathcal{F}$  be a feature extraction method that calculates a global descriptor  $g(o)$  for an image  $o$ , and let  $d_g : \mathcal{F} \times \mathcal{F} \rightarrow \mathbb{R}$  be a distance function between two global descriptor,  $d_g(a, b)$  is a low value (near zero) when descriptors  $a$  and  $b$  are similar and it is a high value when  $a$  and  $b$  are dissimilar. If  $d_g$  satisfies the metric properties, then  $(\mathcal{F}, d_g)$  is a metric space, and the metric space  $(\mathcal{I}, d)$  is defined where  $d(x, y) = d_g(g(x), g(y))$ .

The metric properties represent a tradeoff between efficiency and effectiveness for similarity searches. On one hand, the metric properties enables the use of well studied index structures, accelerating searches by avoiding distance evaluations (as we review in next section). On the other hand, metric properties restrict the similarity model that can be used for comparing two objects [28].

### 2.1.1 Efficiency in metric spaces

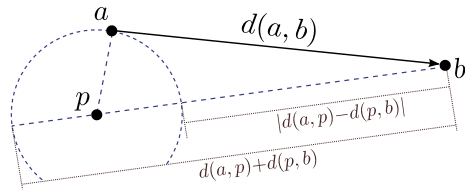
For improving efficiency in metric spaces, the Metric Access Methods (MAMs) [9] are index structures designed to perform efficient similarity search queries. MAMs avoid a sequential scan over the whole database by using the metric properties to save distance evaluations. Given the metric space  $\mathcal{M} = (\mathcal{D}, d)$ , the object-pivot distance constraint [32] guarantees that:

$$\forall a, b, p \in \mathcal{D}, \quad |d(a, p) - d(b, p)| \leq d(a, b) \leq d(a, p) + d(b, p) \quad (1)$$

This constraint implies that for any two objects  $a$  and  $b$ , a lower bound and an upper bound of  $d(a, b)$  can be calculated using a third object  $p$ , which is called a pivot object, see Fig. 1. MAMs group objects in the database and for searching they use (1) to discard groups of objects, thus saving evaluations of  $d$  and search time. MAMs differ in their methods for grouping objects and for selecting pivots. Some examples of MAMs are the M-Tree [10] and the GNAT [6].

The efficiency that some MAM can achieve is related to: a) the amount of distance evaluations that are discarded when it performs a similarity search, and b) the internal cost for deciding whether some distance can be discarded or not. For analyzing the efficiency that any MAM can achieve in some metric space, Chávez

**Fig. 1** The object-pivot distance constraint for distance  $d(a, b)$  using the pivot object  $p$



et al. [9] propose to analyze the histogram of distances. The histogram of distances is constructed by sampling many pairs of objects  $a, b$ , evaluating distances  $d(a, b)$ , and accumulating them into a histogram, see Fig. 2.

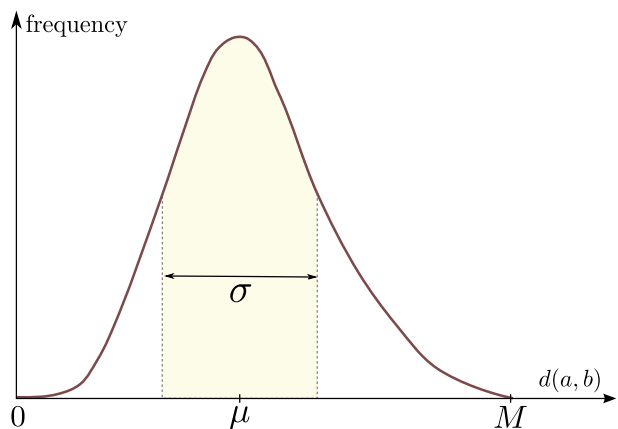
The intrinsic dimensionality  $\rho$  of a metric space is defined as:

$$\rho(\mathcal{M}) = \frac{\mu^2}{2\sigma^2} \quad , \quad (2)$$

where  $\mu$  and  $\sigma^2$  are the mean and the variance of the histogram of distances. The intrinsic dimensionality tries to quantify the difficulty for indexing a metric space. A high  $\rho$  implies that the difference between any two distances will probably be a small value, thus the lower bounds for most of the pivots will be too low to discard objects. Increasing the number of pivots will improve the value of the lower bounds, however the internal cost of the MAM will also increase.

Due to the inherent subjectiveness of image similarity and imprecision of image descriptors, approximate searches can improve the efficiency for Image Retrieval systems with a low cost in the effectiveness. They offer much faster searches at the cost of losing accurateness, i.e. the nearest neighbors that the approximate NN search returns might not be the actual nearest neighbors for the query object. The approaches for approximate searches can be broadly classified into two categories [32]: (1) a reduction of the data to be examined, analyzing less data than is technically needed; and (2) a transformation of the metric space, replacing the domain or the distance of the metric space to reduce the search cost. In this work, we present an approximate search that replaces a costly distance function with a fast estimator.

**Fig. 2** Histogram of distances with median  $\mu$ , variance  $\sigma^2$ , and maximum distance  $M$  for some metric space  $\mathcal{M} = (\mathcal{R}, d)$



### 2.1.2 Effectiveness in metric spaces

For improving effectiveness in metric spaces, one approach is to combine metric spaces defining a dissimilarity function as a linear combination of metrics. Let  $\mathcal{D}$  be the set of objects, let  $\{g_1, \dots, g_m\}$  be a set of a feature extraction methods where  $g_i : \mathcal{D} \rightarrow \mathcal{F}_i$  extracts a global descriptor, let  $\{d_1, \dots, d_m\}$  be a set of distance functions where  $d_i : \mathcal{F}_i \times \mathcal{F}_i \rightarrow \mathbb{R}$  is a dissimilarity function that defines the metric space  $(\mathcal{F}_i, d_i)$ , and let  $\{w_1, \dots, w_m\}$  be the set of weights  $w_i \in \mathbb{R}$ , then a multi-metric space [7]  $(\mathcal{D}, \delta)$  is defined where  $\delta : \mathcal{D} \times \mathcal{D} \rightarrow \mathbb{R}$  calculates the dissimilarity between two objects as:

$$\forall x, y \in \mathcal{D}, \quad \delta(x, y) = \sum_{i=1}^m w_i \cdot d_i(g_i(x), g_i(y))$$

We will call the set  $\{d_1, \dots, d_m\}$  as the underlying metrics of  $\delta$ . In this work, we will focus on convex combinations, i.e.  $w_i \in [0, 1]$  and  $\sum_{i=1}^m w_i = 1$ .

Bustos et al. [7] proposed a dynamic weighting of metrics, called *entropy impurity*, where each weight changes depending on the query object. This method computes the set of weights prior to each similarity search by analyzing the result of a search on a database already classified. A similar technique is also proposed by Deselaers et al. [11] under the name of *maximum entropy*. The major problem with dynamic weighting is that it breaks the metric properties, thus general MAMs cannot be used.

If weights are static (i.e. a fixed value for all searches), then  $\delta$  also satisfies the metric properties [4], thus any MAM can be used for indexing the objects. The value assigned to each weight depends on the actual implementation of descriptors (a different weight is required depending on whether a descriptor represents colors or textures), and application specifics (a system that retrieves sport images may use different weights than a system that retrieves hand-made sketches). The set of weights can be fixed subjectively as fine tuning parameters [4], or can be fixed in accordance with evaluations of effectiveness. However, for some systems the evaluation of the effectiveness might be difficult to define (an evaluation process and good indicators must be chosen), expensive to perform (requiring the hiring of users to test the system and fill out evaluation forms), or even impossible (for corpuses with unknown solution). In this work we present a novel technique for selecting automatically a set of weights with high effectiveness without performing any effectiveness evaluation.

## 2.2 Content-based video copy detection

Content-Based Video Copy Detection (CBVCD) consists of detecting videos that are copies of known original videos, and retrieving the segment of the original video that was copied. The detection method must only make use of visual and audio content, so it cannot embed watermarks or use any metadata in the original videos. Joly et al. [15] propose a definition of “copy” based on a subjective notion of tolerated transformations: A tolerated transformation (TT) is a function that creates a new version of a document where the original document “remains recognizable”. That is, let  $T$  be a set of TTs, and  $u$  and  $v$  be two videos,  $v$  will be a copy of  $u$  if  $\exists t \in T, t(u) = v$ .

Some problems where CBVCD techniques can be applied are: identification of known sequences in video streams (like commercials in a television stream), purging multiple copies of the same video content in a collection, checking for copyright infringements in videos uploaded by users to video sharing sites, grouping of videos in a post-processing phase of a video search engine, detection of commons shots for a video footage managing system, replication of video semantic descriptions, and mining of video databases.

CBVCD systems usually rely on two different tasks: Feature Extraction and Similarity Search. On one hand, the Feature Extraction task calculates representative descriptors for a video sequence. CBVCD systems extract keyframes from each video, and from those keyframes they extract descriptors. Frame descriptors can be either global or local. Global descriptors represent the content of a whole frame. In order to achieve high effectiveness, the descriptors calculated by the feature extraction method should be as invariant as possible to the TTs. Some examples of global descriptors for CBVCD are color histograms [14] and the Ordinal Measurement (OM) [16]. Local descriptors represent the neighborhood of interest points in a frame. In order to achieve high effectiveness, the interest points detection and the local descriptor should be as invariant as possible to the TTs. The most commonly used local descriptors for CBVCD are SIFT [21] and SURF [5].

On the other hand, the Similarity Search task is implemented by an algorithm that finds objects in an indexed collection that match a given query. Three different search approaches are typically followed by CBVCD systems: continuous, discrete, and probabilistic.

First, the continuous approach corresponds to the traditional search in metric spaces. Given a query video and a distance function, it performs a range or a nearest neighbor search, selecting the closest objects to the query. This is the preferred approach for global descriptors. For example, Hampapur et al. [14] compare the effectiveness of different global descriptors which are extracted from keyframes selected by regular sampling. They compare their effectiveness by changing the range threshold of the search. Kim et al. [16] extract an OM descriptor for video keyframes, and a temporal OM with the changes (increase or decrease) of the intensity for consecutive keyframes. The distance between two videos is defined as the combination of a spatial difference and a temporal difference, where the spatial difference is the average of the differences between their OM descriptors, and the temporal difference is the average of differences between their temporal OM descriptors. This distance is compared to a threshold in order to decide the existence of a copy between the two videos. The search algorithm takes every video in the database and calculates the distance to the query video probing all possible alignments without using any index structure. In other work, Gupta et al. [13] extract audio-based descriptors, which are compared by performing nearest neighbor searches implemented on GPU (the searches are sequential scans, without an index).

Second, the discrete approach represents each descriptor with a value taken from a fixed set. It was initially proposed by Sivic and Zisserman [27] and is also known as the bag-of-features approach. A system implementing this approach proved to be highly effective and efficient in TRECVID 2008 [12]. In other work, Poullot et al. [26] create a global descriptor for each frame from its local descriptors, which they call “glocal descriptors”, and organize them in an inverted index for searching similar frames.

Third, the probabilistic approach performs a probability-based approximate search. Joly et al. [15] partition the space with a Hilbert space filling curve and estimate a distribution of descriptors for each block. Poullot et al. [25] replace the Hilbert curves with Z-order curves maintaining the probability-based search method. Law-To et al. [18] track the trajectories of persistent interest points between consecutive keyframes. If the points are static, they represent the background, and if they are moving, they represent objects in motion. A probabilistic search is performed, and with a voting algorithm based on a geometric model they can detect copies even for complex TTs, like replacement of background or insertion of characters in a scene.

In recent years, there has been a lot of research on the video copy detection topic. The main issues currently being faced are the development of new techniques for managing local descriptors, and the development of different fusion techniques for local, global, and audio descriptors.

### 3 CBVCD using metric spaces and global descriptors

In this section, we introduce our approach for implementing a CBVCD system using metric spaces. Let  $\mathcal{V}$  be the set of videos for the reference collection (the known originals), let  $\mathcal{T}$  be the set TTs that can be applied to reference videos, and let  $\mathcal{C}$  be the set of videos for the query collection (the potential copies). For each query video  $c \in \mathcal{C}$ , a CBVCD system returns a list of copy detections  $(\bar{c}, \bar{v}, s)$ , where  $\bar{c}$  is a segment from  $c$ ,  $\bar{v}$  is a segment from an original video  $v \in \mathcal{V}$ , and  $s \in \mathbb{R}^+$  is a confidence score for  $t(\bar{v}) = \bar{c}$  for some  $t \in \mathcal{T}$ .

In our approach, we divide a CBVCD system into five tasks: (1) Preprocessing, (2) Video Segmentation, (3) Feature Extraction, (4) Similarity Search, and (5) Copy Localization. As a general overview, the Preprocessing task processes every query and reference video with the objective of normalizing reference and query videos and diminishing the effect of TTs on query videos. This task creates a new set of query videos  $\mathcal{C}'$  and reference videos  $\mathcal{V}'$ . The Video Segmentation task partitions every video into short video segments, producing a set of query segments and a set of reference segments. The Feature Extraction task calculates many global descriptors for every segment. The Similarity Search task performs a NN+range search for each query segment which returns the  $k$  most similar reference segments. The Copy Localization task takes the sets of similar segments for the segments in a query video, and it searches for groups of reference segments that belong to the same reference video. This task produces a set of detection candidates between preprocessed videos  $(\bar{c}', \bar{v}', s')$ , which then are combined to return the final set of detections  $(\bar{c}, \bar{v}, s)$ .

In the following subsections we review in detail each of these tasks.

#### 3.1 Preprocessing

This task has two objectives: (1) to normalize the quality of videos in  $\mathcal{V}$  and  $\mathcal{C}$ , and (2) to diminish the effect of TTs on videos in  $\mathcal{C}$ .

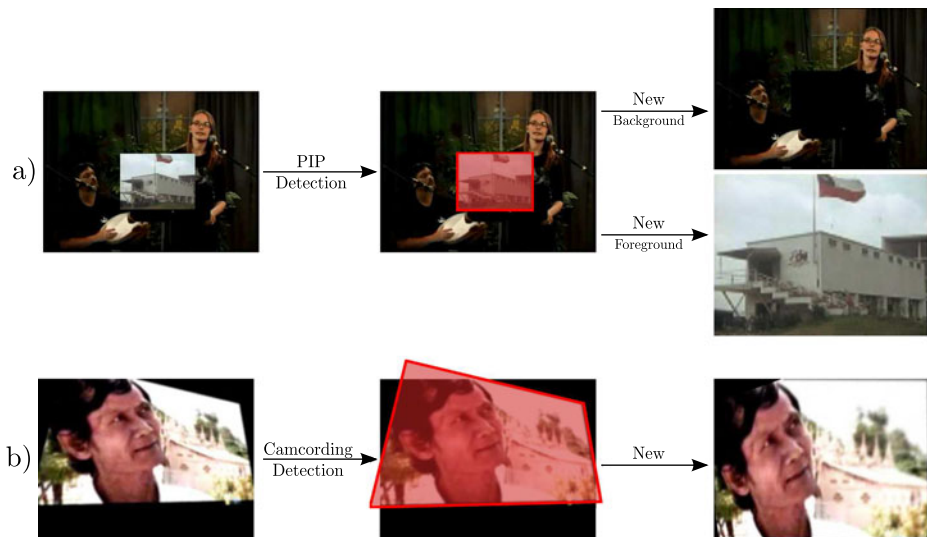
To normalize a video, it processes every frame and marks a frame  $f_i$  to be skipped if it is plain ( $f_i$  has low variance between the intensity of its pixels) or it is an outlier (the image difference between previous frame  $f_{i-1}$  and next frame  $f_{i+1}$  is low and the

image differences between  $f_i$  and  $f_{i-1}$  and between  $f_i$  and  $f_{i+1}$  are high). A frame is skipped by duplicating the previous frame  $f_{i-1}$  when  $i > 0$ , or the next non-skipped frame when  $i = 0$ . Then, it processes the whole video detecting a letter-, pillar-, or window-boxing and removes it by cropping the black borders. For each video  $v \in \mathcal{V}$  its normalized version  $v'$  is added to the output set  $\mathcal{V}'$ , also for each query video  $c \in \mathcal{C}$  its normalized version  $c'$  is added to the output set  $\mathcal{C}'$ .

For diminishing the effect of TTs on a normalized query video  $c'$ , the task requires that the possible TTs in  $\mathcal{T}$  are known. The objective is to create new query videos by detecting and reverting the TTs for which global descriptors are not invariant. In our work, we have focused on preprocessing two TTs: picture-in-picture (PIP) and camcording. If  $\text{PIP} \in \mathcal{T}$ , a PIP detection is performed by detecting a persistent rectangle on  $c'$ . When a PIP is detected two new query videos are created: the foreground video (each frame cropped to the detected rectangle) and the background video (each frame with the detected rectangle filled with black pixels). If  $\text{camcording} \in \mathcal{T}$ , a camcording detection is performed by detecting a wrapping quadrilateral on  $c'$ . When camcording is detected a new query video is created by mapping the detected quadrilateral to the video corners. All the created query videos are normalized and added to the output set  $\mathcal{C}'$ . See Fig. 3.

### 3.2 Video segmentation

The objective of this task is to partition every reference video and query video into short video segments. A short video segment (in the following, a segment) is a group of similar consecutive frames. It is not required that the segments be the same length. Optionally, each segment can select one of its frames as a representative frame. More formally, given a video  $v \in \mathcal{V}' \cup \mathcal{C}'$  of  $n$  frames  $\{f_1, \dots, f_n\}$ , the video segmentation



**Fig. 3** Preprocessing task: **a** query video with PIP, detected rectangle, and new query videos created by the reversion. **b** query video with camcording, detected quadrilateral, and new query video created by the reversion

task partitions  $v$  into a set of  $r$  segments  $\{s_1, \dots, s_r\}$  where  $\forall i \in \{1, \dots, r\}, |s_i| = l_i > 0$ ,  $s_i = \{f_{k_i}, \dots, f_{k_i+l_i-1}\}$ , and  $\forall f \in \{f_1, \dots, f_n\}, \exists! s \in \{s_1, \dots, s_r\}, f \in s$ ; optionally, for each segment  $s \exists! f_s \in s$  that  $f_s$  represents  $s$ .

We divide a video into segments instead of shots, because dividing a video into shots will result in too few segments. Some corpuses contain copies shorter than a single shot, making a denser division necessary. We do not just select keyframes, because for feature extraction we will use a whole segment instead of just representative frames (more details will be given in the next subsection).

In our implementation, we first partition the video into segments of fixed length, and then we join two consecutive segments when all their frames are almost identical between them.

### 3.3 Feature extraction

The objective of this task is to calculate one or more global descriptors to represent each video segment. Let  $\mathcal{S}$  be the set of short video segments, the feature extraction method is defined by the pair  $(g, d)$  where  $g : \mathcal{S} \rightarrow \mathcal{F}$  is the extraction function,  $g(s)$  is the global descriptor for segment  $s$ , and  $d : \mathcal{F} \times \mathcal{F} \rightarrow \mathbb{R}$  is the function that compares descriptors.

This task establishes three requirements for a feature extraction method: (1)  $g(s)$  should represent the whole segment  $s$ ; (2)  $d$  must satisfy the metric properties; (3) the feature extraction method should not be severely affected by the TTs that might have been applied to  $s$ .

The descriptor  $g(s)$  is usually a vector, but it is not required to be the same length for every segment.  $g$  can extract a spatio-temporal descriptor for the whole segment, or it can simply extract a frame-global descriptor for the representative keyframe of the segment. In the last case, however, we extract the frame-global descriptor for every frame in the segment and average the results. Thus, the global descriptor for a segment is the average of the global descriptors for all of its frames. As we will show in the experimental section, the average descriptor implies more computational cost for the feature extraction task, but it improves the effectiveness of the similarity searches.

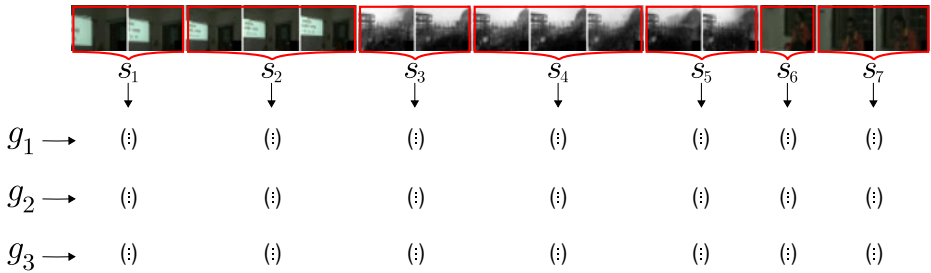
If a feature extraction method is severely affected by some TT, the inclusion of some reversion or normalization for that TT in the preprocessing task should be considered.

Note that adding more global descriptors in our approach is direct. For example, the Feature Extraction task may include a feature extraction method that considers the audio track of the segment, for this  $g$  should implement an acoustic descriptor for the segment, and  $d$  should compare the descriptors with a metric distance.

The output of this task is a set of  $m$  descriptors  $\{g_1(s), \dots, g_m(s)\}$  for each segment  $s$ , and a set of  $m$  metrics  $\{d_1, \dots, d_m\}$  for comparing the descriptors. Figure 4 depicts the Video Segmentation and Feature Extraction tasks for a reference video.

### 3.4 Similarity search

The objective of this task is to perform NN+range searches which retrieve the most similar reference segments for every segment in a query video. For each segment  $q \in \mathcal{Q}$ , the search returns  $N_q$ , which is the list with the  $k$  closest reference segments



**Fig. 4** Video segmentation and feature extraction tasks: the video is partitioned into seven segments by the video segmentation task, and for each segment the feature extraction task calculates three global descriptors

to  $q$  inside a distance threshold  $\epsilon$ , i.e.  $N_q = \{(r^1, dist^1), \dots, (r^k, dist^k)\}$ , where  $r^i \in \mathcal{R}$  is the  $i$ th nearest neighbor at a distance  $dist^i$  to  $q$ , and  $dist^i \leq dist^j$  for  $i \leq j$ .

This task follows three steps: (1) it defines a distance function between two segments, (2) it creates an index structure for efficient evaluation of similarity searches, and (3) it performs an approximate NN+range search for every  $q \in \mathcal{Q}$ .

### 3.4.1 Distance function

Let  $m$  be the number of descriptors extracted for each segment by the Feature Extraction task, let  $g_i(s)$  be the  $i$ th descriptor for segment  $s$ , and let  $d_i$  be the metric for comparing the  $i$ th descriptor,  $i \in \{1, \dots, m\}$ . The spatial dissimilarity between two segments  $q$  and  $r$  is defined as a weighted combination of the distances between its descriptors:

$$\gamma(q, r) = \sum_{i=1}^m w_i \cdot d_i(g_i(q), g_i(r)) \quad (3)$$

where  $w_i \in [0, 1]$  and  $\sum_{i=1}^m w_i = 1$ . The spatio-temporal distance between two video segments is defined as the average dissimilarity between  $W$  consecutive segments:

$$\delta(q_j, r_k) = \frac{1}{W} \sum_{w=1}^W \gamma\left(q_{j+w-\lfloor \frac{W}{2} \rfloor}, r_{k+w-\lfloor \frac{W}{2} \rfloor}\right) \quad (4)$$

where  $W$  is an odd number,  $q_j$  is the  $j$ th segment for a video  $v$  partitioned into segment  $\{q_1, \dots, q_s\}$ ,  $\forall j < 1$   $q_j = q_1$ , and  $\forall j > s$   $q_j = q_s$ . The similarity searches are performed in the metric space  $\mathcal{M} = (\mathcal{D}, \delta)$  where  $\mathcal{D} = \mathcal{R} \cup \mathcal{Q}$ .

Some works including a spatio-temporal distance has been reviewed in Section 2.2. In particular, the temporal distance for the Ordinal Measurement [16] descriptor is closely related, however  $\delta$  is more general because it is not associated with any descriptor.

The first issue that each weight  $w_i$  in (3) should solve is to scale distances returned by each underlying metric  $d_i$  to a range whose values between different metrics are comparable. Usually, these weights are fixed to normalize each distance by its maximum value. This normalization scales every metric to a bounded value in range  $[0, 1]$ , thus they can be combined by  $\delta$ . However, this approach does not reflect the distribution of distances inside each metric, i.e. for some  $d_i$  a distance threshold 0.5

could be a very permissive value (a range search selects many objects) while in other metric  $d_i$  could be a very strict (selecting just a few objects).

To overcome this issue, we set each weight  $w_i$  using the histogram of distances of  $d_i$ . Because the area of the histogram of distances is normalized to 1, then it can be seen as a probability distribution of  $d_i$ . Next, we define the cumulative distribution in a similar way as in probabilities.

**Definition 1** (Cumulative distribution) Let  $d$  be a distance function, and let  $H_d$  be its histogram of distances, the Cumulative Distribution of Distances  $F_d : \mathbb{R}^+ \rightarrow [0, 1]$  is defined as:

$$F_d(x) = \int_0^x H_d(t)dt$$

We state that two distance values are comparable when they have the same selectiveness for their respective distributions, i.e. for two distance functions  $d_1$  and  $d_2$  the values  $d_1(x)$  and  $d_2(y)$  are comparable when  $F_{d_1}(x) \approx F_{d_2}(y)$ .

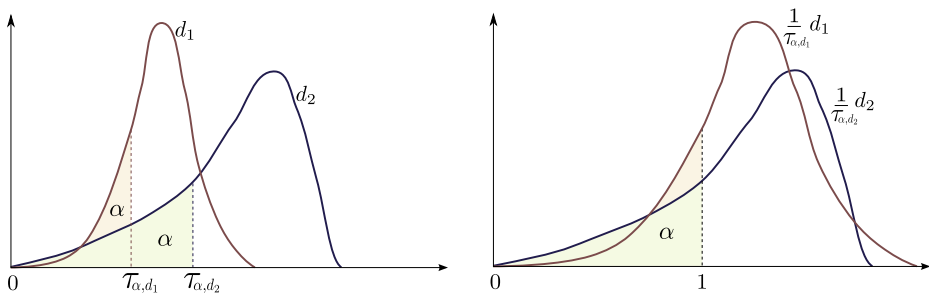
With the objective of scaling distance functions for making their values comparable, we define the  $\alpha$ -normalization.

**Definition 2** ( $\alpha$ -normalization) Let  $d$  be a distance function, and let  $\alpha$  be a number in  $(0,1]$ , the  $\alpha$ -Normalization of  $d$  corresponds to scale distances of  $d$  with the weight:

$$w = \frac{1}{\tau_{\alpha,d}}$$

where  $F_d(\tau_{\alpha,d}) = \alpha$ , and  $F_d$  is the cumulative distribution of distances for  $d$ .

Because  $\delta$  is used to perform NN+range searches, all its underlying metrics should be  $\alpha$ -normalized with a low value. Then, the smaller distances for the underlying metrics will be comparable between them. As a general rule, a value between 0.1 and  $\frac{1}{|\mathcal{R}|}$  is usually good enough. Note that  $\alpha = 1$  implies the normalization by the maximum distance. Note also that  $\alpha$ -normalization with the same  $\alpha$  for different distance functions should imply a different weight for each one. Figure 5 shows the  $\alpha$ -normalization for two distances.



(a) Distance thresholds  $\tau_{\alpha}$  are located for both distances.

(b) Both distances are normalized according to their  $\tau_{\alpha}$ .

**Fig. 5** Example of  $\alpha$ -normalization for two distance functions  $d_1$  and  $d_2$

The  $\alpha$ -normalization tries to select weights that scale the distributions in order to make the smaller distances comparable between them. Because histograms of distances may have different shapes and slopes, the distances become comparable only in a neighborhood of  $\tau_\alpha$ . To make underlying metrics comparable in their whole ranges a variable weight is needed, however in that case the triangle inequality will not be satisfied.

The second issue that weights should support is to give more importance to better descriptors for improving effectiveness. Even when underlying metrics are comparable, depending on the actual descriptors and the dataset, it may be better to increase or decrease some weight on the combination.

The  $\alpha$ -normalization scales each underlying metric  $d_i$  with a weight that for randomly-selected objects  $x, y$ ,  $\mathbb{P}[d_i(x, y) \leq 1] = \alpha$ . Because  $\delta$  is a convex combination of  $\alpha$ -normalized distances, it might be expected that for randomly-selected objects  $x, y$   $\mathbb{P}[\delta(x, y) \leq 1] = \alpha$ . If all the underlying metrics were independent, this statement would be true. However, in general this statement is not true. The reason is that the same pair  $(x, y)$  selected to evaluate  $\delta$  is used for each of its underlying distances. If descriptors  $i$  and  $j$  are highly correlated, then the conditional probability increases  $\mathbb{P}[d_j(x, y) \leq 1 | d_i(x, y) \leq 1] > \alpha$ . For example, if  $d_1$  compares segments by their RGB histogram and  $d_2$  compares them by their HSV histogram, when two segments  $x$  and  $y$  are similar according to the RGB histogram ( $d_1(x, y) \leq 1$ ), the probability of being similar for the HSV histogram will increase ( $\mathbb{P}[d_2(x, y) \leq 1] > \alpha$ ). In that case,  $\delta(x, y)$  will be less than 1 with a probability higher than  $\alpha$ , thus the value of  $\tau_{\alpha, \delta}$  (i.e. the value that  $\alpha$ -normalizes  $\delta$ ) will be smaller than 1. We define the following weighting method that tries to favor uncorrelated descriptors as:

**Definition 3** (Weighting by Max- $\tau$ ) Let  $\delta$  be a convex combination of underlying metrics  $\{d_1, \dots, d_m\}$  with weights  $\{w_1, \dots, w_m\}$ , and let  $\alpha$  be a number in  $(0, 1]$ , the Weighting by Max- $\tau$  first  $\alpha$ -normalizes each underlying metric, and then selects the set of weights that maximizes  $\tau_{\alpha, \delta}$ , where  $\tau_{\alpha, \delta}$  is the value that  $\alpha$ -normalizes  $\delta$ .

For maximization of  $\tau_{\alpha, \delta}$ , the Newton-Raphson method can be used. However, we use a simpler approach that fixes each weight  $w_i = \frac{1}{m}$ , and then iteratively replaces one  $w_i$  with  $w_i \pm \varepsilon$  if that change increases  $\tau_{\alpha, \delta}$ , and ends when every weight has been tested and none updated.

In practice, each underlying metric has two associated weights: an internal weight from its  $\alpha$ -normalization, and an external weight from the Weighting by Max- $\tau$ . Once selected the internal and external weight, both can be multiplied to fix the final weight to each underlying metric.

We should stress that the histogram of distances is created by sampling pairs of objects and not evaluating every possible pair. Depending on the distance function, the evaluation of a statistically-relevant number of distances should take from a few seconds up to a few minutes. Thus, the computational time required for the normalization and weighting algorithms is practically negligible when it is compared to the computational time required for the similarity searches.

### 3.4.2 Pivot-based index

Let  $\mathcal{P} \subseteq \mathcal{R}$  be a set of reference segments, the lower bound function  $LB_{\mathcal{P}}$  for distance  $\delta$  is defined as:

$$LB_{\mathcal{P}}(q, r) = \max_{p \in \mathcal{P}} \{|\delta(q, p) - \delta(r, p)|\} \quad (5)$$

Each object  $p \in \mathcal{P}$  is called a *pivot*. Because  $\delta$  defines the metric space  $\mathcal{M} = (\mathcal{D}, \delta)$ , the object-pivot distance constraint from (1) implies that:

$$\forall q, r \in \mathcal{D}, \forall \mathcal{P} \subseteq \mathcal{R}, LB_{\mathcal{P}}(q, r) \leq \delta(q, r) \quad (6)$$

Note that if  $\delta(x, p)$  is precalculated  $\forall x \in \mathcal{D}$ , then the evaluation of  $LB_{\mathcal{P}}$  costs just  $|\mathcal{P}|$  operations. The index structure consists of a  $|\mathcal{P}| \times |\mathcal{R}|$  table with distances from each pivot to every reference segment. Additionally, for efficient calculation of  $\delta$ , each segment must have a reference to the previous and the next segment in the partition of the video.

A naive approach for selecting pivots is to select objects randomly in  $\mathcal{R}$ . However, a key property for good pivot selection is that pivots should be far away from each other [32]. Sparse Spatial Selection (SSS) [8] uses this property for selecting pivots incrementally, see Algorithm 1. It first randomizes  $\mathcal{R}$  and then it selects sparse pivots depending on the distance that all pivots should be from each other. Note that this algorithm returns a variable number of pivots depending on randomization of  $\mathcal{R}$  and the sparse threshold  $t$ .

---

**Algorithm 1:** Sparse Spatial Selection [7] algorithm for selecting pivots.

---

**Input:**  $\mathcal{R}$  set of reference segments,  $t$  sparse threshold.

**Output:**  $\mathcal{P}$  set of pivots

$\{r_0, \dots, r_{|\mathcal{R}|}\} \leftarrow$  randomize objects in  $\mathcal{R}$

$\mathcal{P} \leftarrow \{r_0\}$

**foreach**  $r_i \in \{r_1, \dots, r_{|\mathcal{R}|}\}$  **do**  
  **if**  $\forall p \in \mathcal{P}, \delta(r_i, p) \geq t$  **then**  
     $\mathcal{P} \leftarrow \mathcal{P} \cup \{r_i\}$

**return**  $\mathcal{P}$

---

Because Algorithm 1 depends on randomization, it may produce many different sets of pivots. The obtained sets of pivots must be evaluated with the objective of selecting the set that will produce tighter values between  $LB_{\mathcal{P}}$  and  $\delta$  in the similarity searches. Given two sets of pivots  $\mathcal{P}_1$  and  $\mathcal{P}_2$  from Algorithm 1, the selection of the best set follows two criteria: a) if  $|\mathcal{P}_1| \neq |\mathcal{P}_2|$  then select the set with more pivots; b) if  $|\mathcal{P}_1| = |\mathcal{P}_2|$  then select the set that produces the higher average value of  $LB_{\mathcal{P}}$ . The first criterion says that more pivots implies better  $\mathcal{P}$ , but the computational cost for evaluating  $LB_{\mathcal{P}}$  depends directly on  $|\mathcal{P}|$ . Therefore, a target number of pivots  $P$  must be chosen. Algorithm 1 is then performed many times with decreasing values for the sparse threshold  $t$  until  $N$  sets of pivots have been selected. Finally, the  $N$  sets are evaluated with the two previous criteria and the best set  $\mathcal{P}$  is selected to create the index.

### 3.4.3 Approximate search with pivots

For every query segment  $q \in \mathcal{Q}$ , the first step is to calculate  $\delta(q, p) \forall p \in \mathcal{P}$ , and then to perform the NN+range search for selecting the  $k$  closest objects inside a distance threshold  $\epsilon$ . Algorithm 2 depicts a classic “exact” algorithm for NN+range search using pivots. It uses (6) to evaluate  $\delta$  only when the lower bound is less than both the range  $\epsilon$  and the  $k$ th candidate distance. Though Algorithm 2 can achieve a big improvement compared to a sequential scan over  $\mathcal{R}$  (and returning the same result), it might not be fast enough for video copy detection databases.

---

#### Algorithm 2: Classic NN+range search using pivots.

---

**Input:**  $q$  query segment,  $\mathcal{R}$  set of reference segments,  $k$  number for NN search,  $\epsilon$  threshold for range search,  $\mathcal{P}$  set of pivots.  
**Output:** List of  $k$  nearest neighbors to  $q$

```

NNs  $\leftarrow$  new priority queue
foreach  $r \in \mathcal{R}$  do
   $lb \leftarrow LB_{\mathcal{P}}(q, r)$  // Equation 5
  if  $lb < \epsilon$  and ( size of NNs  $< k$  or
     $lb < \max$  distance in NNs ) then // Equation 6
     $dist \leftarrow \delta(q, r)$  // Equation 4
    if  $dist < \epsilon$  then
      add  $r$  to NNs with distance  $dist$ 
      if size of NNs  $> k$  then
        remove max distance object from NNs
return NNs

```

---

Algorithm 3, on the other hand, returns an approximation of the exact result of Algorithm 2. It is based on the fact that the lower bounds of the nearest neighbors are usually between the smallest lower bounds. Thus, it uses  $LB_{\mathcal{P}}$  as an estimator for the actual distance: it evaluates  $LB_{\mathcal{P}}$  for every object, it discards objects with  $LB_{\mathcal{P}}$  greater than threshold  $\epsilon$ , it selects the  $T$  objects with smallest values of  $LB_{\mathcal{P}}$ , and just for them it evaluates  $\delta$ . Finally, comparing the  $T$  evaluated distances, it selects the  $k$  nearest neighbors that are closer than  $\epsilon$  to  $q$ . This is an approximate search because there is no guarantee that the  $LB_{\mathcal{P}}$  for the actual NN will be between the  $T$  smallest values of  $LB_{\mathcal{P}}$ .

The key difference between classic search and the approximate search is that while Algorithm 2 uses  $LB_{\mathcal{P}}$  value as a lower bound for discarding objects that will have a high value for  $\delta$ , Algorithm 3 compares  $LB_{\mathcal{P}}$  values between them, assuming that a low/high value for  $LB_{\mathcal{P}}$  implies a low/high value for  $\delta$ . Note that (6) is not used on Algorithm 3 because  $LB_{\mathcal{P}}$  is used just as a fast  $\delta$  estimator and not as a lower bound.

As said in the previous subsection, selecting more pivots for  $\mathcal{P}$  implies tighter estimations, but it also implies a higher computational cost for evaluating  $LB_{\mathcal{P}}$ . However, with tighter estimations a smaller  $T$  is necessary for selecting actual NNs. The tradeoff between  $|\mathcal{P}|$  and  $T$  is analyzed in the experimental section. Note that as  $T$  moves to  $|\mathcal{R}|$ , Algorithm 3 will produce the same results of Algorithm 2 independent of  $\mathcal{P}$ . In particular, when  $T=|\mathcal{R}|$  the approximate search will evaluate  $\delta$  for the whole reference collection, thus it will select the same nearest neighbors as the exact search (but at a higher cost).

**Algorithm 3:** Approximate NN+range search using pivots.

---

**Input:**  $q$  query segment,  $\mathcal{R}$  set of reference segments,  $k$  number for NN search,  $\epsilon$  threshold for range search,  $\mathcal{P}$  set of pivots,  $T$  number of lower bounds.

**Output:** List of approximate  $k$  nearest neighbors to  $q$

MinLbs  $\leftarrow$  new priority queue

**foreach**  $r \in \mathcal{R}$  **do**

lb  $\leftarrow LB_{\mathcal{P}}(q, r)$  // Equation 5

**if** lb  $< \epsilon$  **then**

add  $r$  to MinLbs with distance lb

**if** size of MinLbs  $> T$  **then**

remove max distance object from MinLbs

NNs  $\leftarrow$  new priority queue

**foreach**  $r \in$  MinLbs **do**

dist  $\leftarrow \delta(q, r)$  // Equation 4

**if** dist  $< \epsilon$  **then**

add  $r$  to NNs with distance dist

**if** size of NNs  $> k$  **then**

remove max distance object from NNs

**return** NNs

---

In preliminar experiments we tested more complex estimators. In particular we tested the minimum upper bound, the average bound (the average between lower and upper bounds), and the median of the average bound for all the pivots. However, these estimators did not worth their higher computational cost compared with  $LB_{\mathcal{P}}$ .

### 3.5 Copy localization

The objective of this task is to decide the existence of a copy for each query video  $v$  by analyzing the result of the Similarity Search task. The localization is based on searching groups of nearest neighbors belonging to the same reference video and offset.

The input for this task is: a query video  $c' \in \mathcal{C}'$  partitioned into the segments  $\{s_1, \dots, s_r\}$ , a set  $\{N_1, \dots, N_r\}$  where  $N_i = \{(r_i^1, dist_i^1), \dots, (r_i^k, dist_i^k)\}$  is the list of the nearest neighbors for query segment  $s_i$ ,  $r_i^j \in \mathcal{R}$  is the  $j$ th nearest neighbor,  $dist_i^j \in [0, \epsilon]$  is the distance  $\delta(s_i, r_i^j)$ ,  $dist_i^j \leq dist_i^{j+1} \forall i \in \{1, \dots, r\}$ ,  $j \in \{1, \dots, k\}$ , and  $k$  and  $\epsilon$  are the parameters for the NN+range searches.

This task creates a list of copy detections candidates, where each candidate is composed of the copy bounds (start/end time in query video), reference video, offset, and detection score. An offset is the time that needs to be added to query video bounds for getting reference video bounds, i.e.,  $copy\ start + offset = reference\ start$  and  $copy\ end + offset = reference\ end$ .

The first step compares every  $s_i$  with its neighbors in  $N_i$  collecting all the reference video  $v' \in \mathcal{V}$  that owns some segment  $r_i^j$  and all the minimum and maximum offsets. The offsets are obtained by comparing start/end times for  $s_i$  with start/end times for  $r_i^j$ .

The second step takes every candidate reference video, divides its minimum and maximum offsets into small intervals of fixed length (for example, 0.1 s). A copy candidate is composed of a reference video and an offset interval. Then, it performs a copy detection by executing Algorithm 4 for every copy candidate.

---

**Algorithm 4:** Copy detection algorithm.
 

---

**Input:**  $\{(s_1, N_1), \dots, (s_r, N_r)\}$  set of query segment with its nearest neighbors,  $v'$  reference video candidate, **Offset** offset interval candidate.  
**Output:** start, end and score for copy detection on video  $v'$  with offset **Offset**

```

(start, end, score)  $\leftarrow$  (null, null, 0)
(cstart, cend, cscore)  $\leftarrow$  (null, null, 0)
foreach  $(s_i, N_i) \in \{(s_1, N_1), \dots, (s_r, N_r)\}$  do
  (voter, vote)  $\leftarrow$  CalculateVote( $s_i, N_i, v', \text{Offset}$ )
  cscore  $\leftarrow$  cscore + vote
  if cscore < 0 then
    (cstart, cend, cscore)  $\leftarrow$  (null, null, 0)
  else if vote > 0 then
    if cstart is null then
      cstart  $\leftarrow$  voter
    cend  $\leftarrow$  voter
    if cscore > score then
      (start, end, score)  $\leftarrow$  (cstart, cend, cscore)
return (start, end, score)
  
```

---

For each copy candidate (reference video and offset), Algorithm 4 returns the query video bounds (start/end) and the copy detection score, thus it produces a list of detections  $(\bar{c}', \bar{v}', s')$ . The detection score is the sum of votes that received that copy candidate from reference segments in the lists of nearest neighbors. Each vote and voter segment are calculated by the function *CalculateVote*, depicted in Algorithm 5.

---

**Algorithm 5:** *CalculateVote* function.
 

---

**Input:**  $s$  query segment,  $\{(r^1, dist^1), \dots, (r^k, dist^k)\}$  list of the  $k$  nearest neighbors to  $s$ ,  $v'$  reference video candidate, **Offset** offset interval candidate.  
**Output:** voter best matching segment, vote score of match.

```

(voter, vote)  $\leftarrow$  (null, MissCost)
foreach  $(r^j, dist^j) \in \{(r^1, dist^1), \dots, (r^k, dist^k)\}$  do
  if  $r^j \in v'$  and  $offset(s, r^j) \in \text{Offset}$  then
     $v \leftarrow MatchVote$ 
       $\times$  relevance of distance  $dist^j$  between  $[0, \epsilon]$ 
       $\times$  relevance of rank  $j$  between  $[1, k]$ 
    if  $v > vote$  then
      (voter, vote)  $\leftarrow$  ( $r^j, v$ )
return (voter, vote)
  
```

---

In Algorithm 5, *MatchVote* is the value for a supporting vote (for example, 1), which is weighted according to the relevance of the distance and rank of the voter segment. The relevance of a distance is a value near 0 when distance is  $\epsilon$ , and it is 1 when the distance is 0, we use de cumulative distribution of  $\delta$  to evaluate  $(F_\delta(\epsilon) - F_\delta(dist^j))/F_\delta(\epsilon)$ . The relevance of a rank is a value near 0 when  $j = k$ , and it is 1

when  $j = 1$ . *MissCost* is the cost when there is not a reference segment supporting the detection. This value should be zero or a negative for favoring detections without discontinuities (for example,  $-0.1$ ).

The third and last step combines the list of detections for the query videos created by the Preprocessing task. For each original query video  $c \in \mathcal{C}$ , it searches all its derived videos  $c' \in \mathcal{C}$  in the list of detections and combines them. It eliminates eventual overlaps between detections by keeping the detection with a higher score. Additionally, it may adjust the scores comparing the best score with the second best, or it may scale them to the interval  $[0,1]$ . Finally, it reports the final list of detections  $(\bar{c}, \bar{v}, s)$ .

## 4 Experimental evaluation

To assess the impact of the different tasks on effectiveness and efficiency, a series of experiments were performed and the results are presented below.

### 4.1 Preliminaries

Two video collections were used for the experiments: MUSCLE-VCD-2007 [17] and TRECVID 2010 [29]. The experiments on the former collection measured the behavior of P-VCD in detail, while the experiments with the latter compared its performance with other state-of-the-art CBVCD systems.

#### 4.1.1 Global descriptors

For the next experiments, eight global descriptors were used:

- **Edge Histogram** [22]. It converts an input frame to gray scale, partitions it into  $N_w \times N_h$  sub-images, divides each sub-image into  $M_w \times M_h$  blocks, further partitions each block into  $2 \times 2$  sub-blocks, computes the average intensity of the pixels for each sub-block, and applies an edge detector to each block. The edge detector is comprised of  $b$  different filters of  $2 \times 2$  pixels. If the filter with maximum strength exceeds a certain threshold  $t$ , the block is marked as an edge block. A histogram of edge blocks is created for each sub-image, thus the final vector has  $N_w \times N_h \times b$  dimensions. For this work, we set  $N_w = N_h = 4$ ,  $M_w = M_h = 8$ , and  $t = 5$ . We test two descriptors: **EH5** uses the set of five filters in [22] ( $b = 5$ ) producing a vector of 80 dimensions; and **EH10** uses the same filters and the opposite versions ( $b = 10$ ) producing a vector of 160 dimensions for EH10.
- **Gray Histogram**. Converts a frame to gray scale, divides it into  $N_w \times N_h$  blocks, and for each block calculates a histogram of  $M$  bins. We tested two descriptors: **G14** where  $N_w = 1$ ,  $N_h = 4$ ,  $M = 32$  producing a vector of 128 dimensions; and **G44** where  $N_w = N_h = 4$ ,  $M = 10$  producing a vector of 160 dimensions.
- **RGB Histogram**. Divides a frame into  $N_w \times N_h$  blocks, and for each block calculates a histogram of  $M$  bins for each Red, Green, and Blue channel. We tested two descriptors: **C14** where  $N_w = 1$ ,  $N_h = 4$ ,  $M = 16$ ; and **C44** where  $N_w = N_h = M = 4$ ; both producing a vector of 192 dimensions.

- **Reduced Image.** Converts a frame to gray scale, scales it down to  $N_w \times N_h$  pixels, and creates a vector of  $N_w \times N_h$  dimensions whose values contains the pixel intensities. We tested the descriptor **K11** where  $N_w = 11$  and  $N_h = 9$ , producing a vector of 99 dimensions.
- **Ordinal Measurement [16].** Converts a frame to gray scale, divides it into  $N_w \times N_h$  blocks, and for each block it calculates the average intensity. Then, the average intensities are sorted in ascending order, and the final descriptor corresponds to the rank assigned to each block. We tested the descriptor **OM9** where  $N_w = N_h = 9$ , producing a vector of 81 dimensions.

Each value in the extracted vectors were uniformly quantized into a 8-bit value, i.e. an integer value between 0 and 255. Between the eight tested descriptors, only G14 and C14 are invariant to vertical mirroring. We chose to use the same  $L_1$  distance (Manhattan distance) as the metric for comparing all descriptors:

$$L_1((x_1, \dots, x_n), (y_1, \dots, y_n)) = \sum_{i=1}^n |x_i - y_i|$$

Although we could use a different metric for comparing each of the eight descriptors, we decided to use only  $L_1$  for simplifying the analysis. Moreover,  $L_1$  is very fast to evaluate, it satisfies the metric properties, and it achieves a satisfactory effectiveness for all the eight tested descriptor.

#### 4.1.2 MUSCLE-VCD-2007 dataset

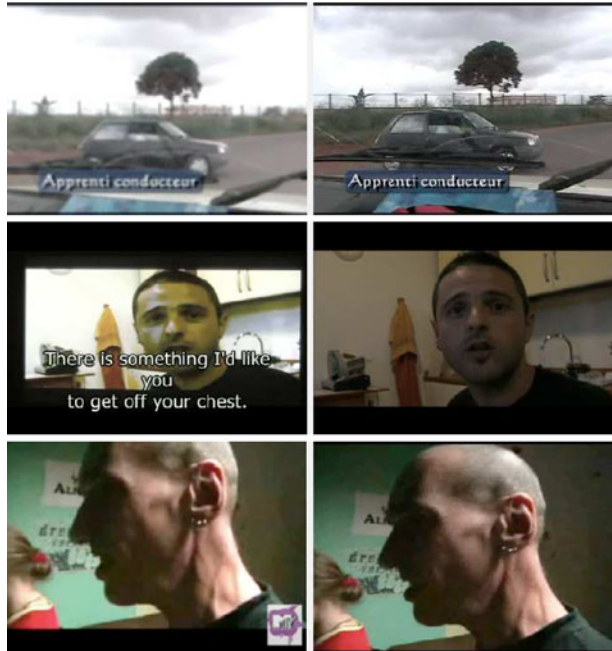
MUSCLE-VCD-2007 is the corpus used for CIVR 2007 video copy detection evaluation. Currently, MUSCLE is a publicly available and widely-used video copy database. The reference collection is composed of 101 videos, 60 h total length (5,285,921 frames, each video at 25 fps), and the query collection (ST1 and ST2 sets) is composed of 18 query videos (15 from ST1 and 3 from ST2), 4 h total length (348,321 frames, each video at 25 fps). Query videos contain 31 segments (10 in ST1 and 21 in ST2) extracted from different reference videos mixed with segments from videos not in the reference collection. Approximately 3 h of query videos proceed from reference collection. Each segment in query videos may have some TT. The possible TTs are a combination of blur, mirroring, subtitles, camcording, zoom, change in brightness and contrast, insertion of logo, noise, and change in color. See Fig. 6.

Between the public databases for video copy detection, we have chosen the MUSCLE-VCD-2007 database because it has a well defined ground truth (it is possible to suit a frame-to-frame match between query videos and reference videos), and its size is appropriate for the following evaluations. It is big enough to be a challenging database, and exact searches can be resolved in reasonable time, thus it is possible to evaluate the real effectiveness of approximate searches.

#### 4.1.3 Evaluation measures

As in previous sections, we define  $\mathcal{R}$  as the set of reference segments, and  $\mathcal{Q}$  as the set of query segments. A similarity search is performed for each  $q \in \mathcal{Q}$  using distance  $\delta$  on  $\mathcal{R}$ . We stated that the correct answer for segment  $q$  with representative frame  $f_q$

**Fig. 6** Some examples of video copies in MUSCLE-VCD-2007. On the left a query video and on the right the corresponding reference video



is the reference segment which contains the original frame for  $f_q$ , i.e., is the segment  $r$  which  $t(x)=f_q$  for some frame  $x \in r$  and some  $t \in \mathcal{T}$ .

Each experiment first fixes some configuration for segmentation, descriptors, distances, and weights, then the similarity searches are performed. For evaluating the Similarity Search task, the rank of the correct answer for each segment  $q$  is calculated. The best value for the rank  $R_q$  is 1 and the worst value is  $|\mathcal{R}|$ . Once performed the  $|\mathcal{Q}|$  similarity searches, we calculate the following effectiveness measures:

- Mean average precision (MAP):  $\frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \frac{1}{R_q}$ . MAP values range between 0 and 1, where 1 means best effectiveness. In these experiments, MAP corresponds to the inverse of harmonic mean of  $R_q$ , thus it is mainly influenced by the smaller ranks mitigating the impact of large outliers.
- Amount of queries with correct answer between the first  $k$  nearest neighbors ( $Nk$ ):  $\frac{|C_k|}{|\mathcal{Q}|}$  where  $C_k = \{q \in \mathcal{Q}, R_q \leq k\}$ .  $Nk$  values range between 0 and 1, where 1 means best effectiveness. This measure is calculated for  $k \in \{1, 10\}$ .
- Number of detected copies without false alarms: First, the Similarity Search task performs a 5-NN search for each query segment. Then, the Copy Localization task performs a copy detection ( $MatchVote = 1$ ,  $MissCost = -0.1$ ). Finally, the copy candidates are sorted by their scores, and the correct candidates with a score greater than the score of the first incorrect candidate are counted, i.e. the number of true positives without false positives. We stated that a copy candidate is correct when its query and reference extents intersect the actual copy extents. The maximum value for correct detections is 31 (10 copies in ST1 and 21 copies in ST2).

## 4.2 Experiments on MUSCLE-VCD-2007

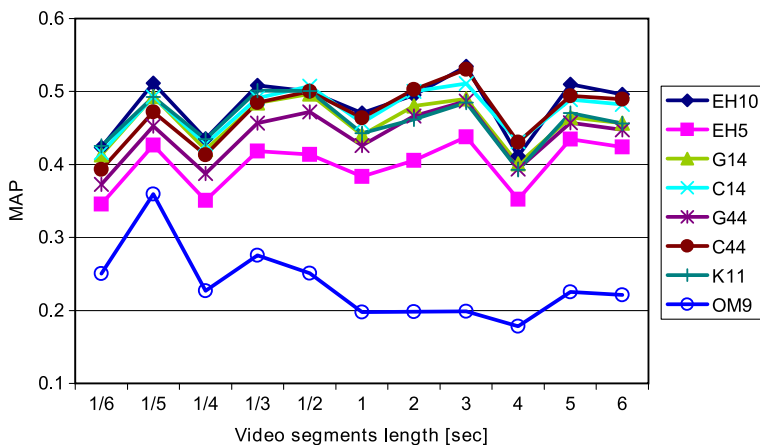
In the following experiments,  $g_i$  is the  $i$ th feature extraction method that can calculate global descriptors for either a video segment or a frame,  $g_i(s)$  is the  $i$ th global descriptor for video segment  $s$ , and  $g_i(f)$  is the  $i$ th global descriptor for frame  $f$ .

### 4.2.1 Effectiveness vs segmentation

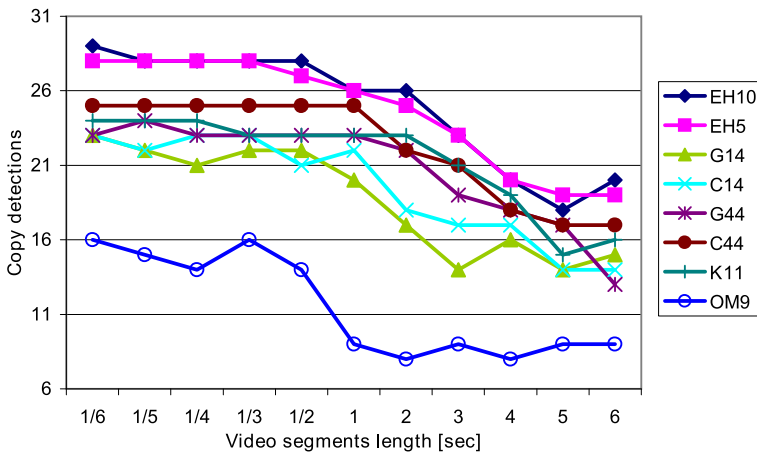
The first experiment tests the effectiveness of each of the eight descriptors varying the size of the video segments. Each query video and reference video is uniformly partitioned into segments of fixed length  $\{\frac{1}{6}, \frac{1}{5}, \frac{1}{4}, \frac{1}{3}, \frac{1}{2}, 1, 2, 3, 4, 5, 6\}$  seconds. The amount of query segments that have a correct answer in the reference collection are 60k, 50k, 40k, 30k, 20k, 10k, 5k, 3.3k, 2.5k, 2k, and 1.7k segments, respectively for each segment length. In this experiment, the descriptor for a segment is just the descriptor for its middle frame (keyframe).

Figure 7 summarizes the effectiveness of the similarity searches for each segment size and for each descriptor. It shows that the effectiveness of the similarity search for each descriptor is relatively independent of the segment size. A coarse segmentation, on one hand, reduces the number of objects in  $\mathcal{Q}$  and  $\mathcal{R}$  which implies faster similarity searches (e.g. doubling the length of the segments divides by a half the number of queries to perform and the reference collection to search in). On the other hand, a coarse segmentation affects the performance of the Copy Localization task due to the decrease in the query segments that can vote (e.g. a video copy of 4 s length can have 20 potential voters with a segmentation of  $\frac{1}{5}$ , but with a segmentation of two seconds length it will only have two potential voters).

Figure 8 shows the performance of the Copy Localization task for the different segmentations and descriptors. Even though a coarse segmentation may achieve a high MAP for the searches, the detection performance is low mainly due to the reduction of voter segments. A fine segmentation can achieve higher detection performance however it can eventually produce false positives with high score.



**Fig. 7** Effectiveness of each descriptor for different video segmentations



**Fig. 8** Detected copies without false alarms for each descriptor and segmentation

The performance varies for each descriptor, however it is clear that OM9 achieves the worst overall result. Ordinal Measurement is invariant to some global changes (like blurring or brightness), but it is highly affected by partial changes that can convert a dark zone into a bright one (like the insertion of logo), changing the order of zones and thus affecting the whole descriptor. This behavior for the Ordinal Measurement can also be verified in [19]. The other evaluated descriptors also divide the frames, but a change in one zone does not affect the values for the other zones.

An interesting result is that EH10 descriptor with segmentation  $\frac{1}{6}$  detects correctly 29 of 31 copies, i.e. it achieves precision 1 and recall 93.5%.<sup>1</sup> This result outperforms most of the state-of-the-art systems evaluated with MUSCLE-VCD-2007 corpus, like a complex system based on local descriptor [26] and a system combining visual with audio descriptors [1].

In general, a segmentation between  $\frac{1}{3}$  and  $\frac{1}{2}$  seconds length achieves a satisfactory balance between detection effectiveness and search time. For the TRECVID evaluation we chose a variable segmentation, with a basal size of  $\frac{1}{3}$  seconds and a postprocessing that joined almost identical consecutive segments [2].

#### 4.2.2 Effectiveness vs average descriptor and temporal window

In the next experiment we fix the video segmentation to one second. This experiment tests: a) the performance of the average descriptor for a segment, defined as the average of the global descriptors for all of its frames (see Section 3.3); and b) the performance of increasing the temporal window to  $W=3$  in  $\delta$  (see (4)).

Table 1 shows the effectiveness of the eight descriptors in four configurations: the first column shows the MAP of comparing segments by the global descriptors of their keyframes (already shown in previous experiment), the second column shows

<sup>1</sup>In particular, the system achieves recall 1 for ST1 and recall 90.5% for ST2. Despite EH10 descriptor is not invariant to mirroring, the system can detect a flipped copy from ST1 by just matching the mostly symmetrical frames.

**Table 1** MAP for keyframe descriptor and average global descriptor with  $W = 1$  and  $W = 3$  (segments of one second length)

	Keyframe $W = 1$	Average $W = 1$	Keyframe $W = 3$	Average $W = 3$
EH10	0.471	0.664	0.643	0.810
EH5	0.383	0.600	0.564	0.760
G14	0.440	0.489	0.529	0.602
C14	0.457	0.501	0.542	0.611
G44	0.426	0.485	0.541	0.610
C44	0.464	0.504	0.589	0.654
K11	0.443	0.510	0.564	0.670
OM9	0.198	0.341	0.281	0.507

the MAP of comparing segments by the average descriptor of their frames, the third column shows the MAP of comparing segments by their keyframes and setting  $W=3$  in  $\delta$ , and the fourth column shows the MAP of comparing segments by their average descriptor and setting  $W=3$  in  $\delta$ .

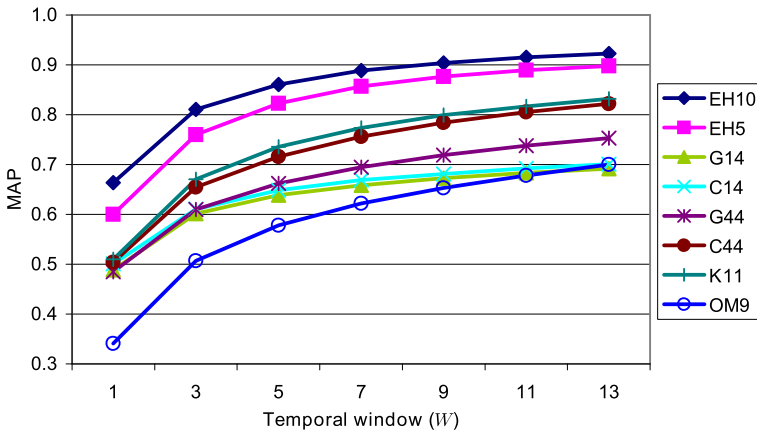
The changes in the effectiveness of the searches show that: a) the effectiveness of each descriptor can be improved by averaging the descriptor of all the frames in a segment (it increases MAP by nearly a 9% for C14 and C44 up to more than 50% for EH5 and OM9); b) the effectiveness of each descriptor can also be improved by averaging the distance of the previous and the next segment (it increases MAP by nearly a 20% for C14 and G14 up to more than 40% for EH5 and OM9); c) both increases are independent and can be combined for greatly improving the effectiveness of each descriptor (it increases MAP by nearly a 33% for C14, and by nearly a 100% for EH5).

Additionally, we also tested the effectiveness of extracting a descriptor from the average frame of a segment. This kind of descriptor always achieved a lower MAP than the descriptor for the keyframe (for every descriptor, for different video segmentations, and for different values of  $W$ ), mainly because average frames usually have grayed colors and blurred edges where descriptor can not be discriminant enough. Thus, it is better to average the descriptors extracted from each frame in a segment than to extract the descriptor from the average frame.

The improvement due to the temporal window of the  $\delta$  function induces us to test the performance of even larger values for  $W$ . Figure 9 shows the improvements for each average descriptor when  $W=\{1, 3, 5, 7, 9, 11, 13\}$ . Each descriptor increases its effectiveness in almost a similar proportion, thus larger  $W$  can improve the effectiveness of the similarity search for every descriptor. The experiment uses segments of one second length, but it should be noted that with different segment lengths the behavior is similar.

Figure 10 shows the detection performance of using the average descriptor and increasing the temporal window. The results for  $W=1$  show that the average descriptor improves the detection performance for EH10 and EH5 (they both achieve 29 of 31 detections using segments of one second length instead of  $\frac{1}{6}$  from Fig. 8). For larger  $W$ , the detection performance does not increase as well as the similarity search MAP. Then, large temporal windows are mainly useful for improving the localization and the detection score of copies already detectable by small windows.

A large temporal window, on the other hand, implies more computational effort. An increase of  $W$  from 1 to 11 implies 10 times more evaluations of  $\gamma$  function. In an exact search, where every query segment is compared to every reference

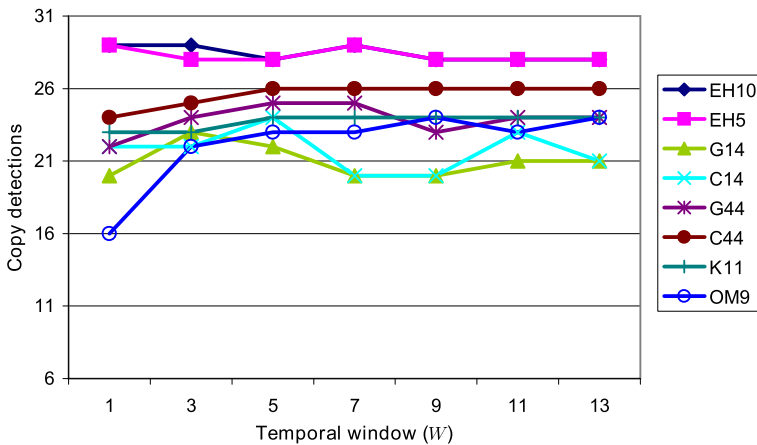


**Fig. 9** Effectiveness of increasing temporal window ( $W$ ) for each average descriptor (segments of one second length)

segment, most of these evaluations are repeated and some optimization can be used (like memoization). For example, in the case of  $W = 3$ , the evaluation of  $\delta(q_i, r_j)$  and  $\delta(q_{i+1}, r_{j+1})$  shares the evaluation of  $\gamma(q_i, r_j)$  and  $\gamma(q_{i+1}, r_{j+1})$ . However, this is not the case for approximate searches where most of the evaluations of  $\delta$  are only estimated and not evaluated. In our system, we chose  $W = 3$  due to its satisfactory tradeoff between computational effort and effectiveness improvement.

#### 4.2.3 Effectiveness vs weighting

The next experiments test the combination of the average descriptors with  $W=1$  and segments of one second length. The MAP for each single descriptor is given in the second column of Table 1.



**Fig. 10** Detected copies without false alarms for the average descriptor and increasing temporal window  $W$  (segments of one second length)

Table 2 compares the effectiveness of combining two and three descriptors using the Normalization by Maximum Distance, the  $\alpha$ -Normalization, and the Weighting by Max- $\tau$ . It shows that Weighting by Max- $\tau$  improves the performance of  $\alpha$ -Normalization, and both outperform the Normalization by Maximum Distance. The best effectiveness was achieved for  $\alpha=0.1$ .

Figure 11 shows the MAP performance for combining between 1 up to 8 descriptors in the following arbitrary order: EH10, C14, K11, G14, G44, C44, OM9, and EH5. The combination normalizes distances with  $\alpha = 0.1$ . It compares the effectiveness of combining average descriptor (AvgD) and keyframe descriptor (KfD) using the  $\alpha$ -Normalization and the Weighting by Max- $\tau$ . It shows that combining descriptors improves the effectiveness up to a saturation point (in this case, three descriptors), and then adding more descriptors may even reduce the effectiveness. As shown in previous experiments, the average descriptor outperforms the descriptor for the keyframe. The Weighting by Max- $\tau$  automatically selects a good set of weights that avoids the decrease in effectiveness due to saturation.

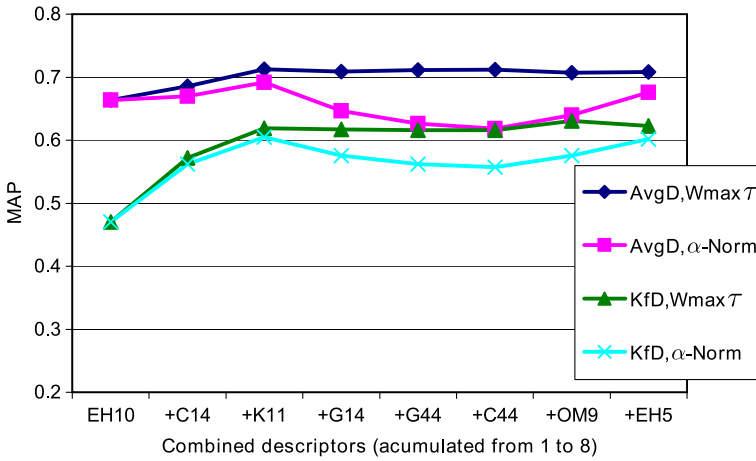
Table 2 and Fig. 11 show that the Weighting by Max- $\tau$  selects automatically a set of weights with high MAP. However, it does not necessarily select the set that achieves the highest MAP. To analyze the relationship between MAP and  $\tau_{\alpha,\delta}$ , we calculated them both for different sets of weights. Table 3 shows the MAP achieved by the combination of descriptors EH10 with C14 and descriptors C44 with K11 depending on the  $\alpha$ -normalization and weights.

Table 3a shows that the highest MAP for EH10+C14 is reached with weights around (0.8, 0.2). The maximum  $\tau_{\alpha,\delta}$  is reached with weights around (0.6, 0.4). Then, the Weighting by Max- $\tau$  can correct the trivial weights (0.5, 0.5) to a set of weights with higher MAP. Although, in this case, the weights will be corrected only to (0.55, 0.45).

**Table 2** Effectiveness of combining two and three descriptors (segments of one second length,  $W = 1$ )

Combination of EH10 and C14					
$\alpha$	$w_{EH10}$	$w_{C14}$	$\tau_{\alpha,\delta}$	MAP	
1	0.5	0.5	–	0.636	
0.1	0.5	0.5	1.066	0.669	
0.1	0.55	0.45	1.066	<b>0.686</b>	
0.01	0.5	0.5	1.148	0.664	
0.01	0.55	0.45	1.150	0.681	
0.001	0.5	0.5	1.238	0.656	
0.001	0.55	0.45	1.241	0.674	
Combination of EH10, C14 and K11					
$\alpha$	$w_{EH10}$	$w_{C14}$	$w_{K11}$	$\tau_{\alpha,\delta}$	MAP
1	0.333	0.333	0.333	–	0.671
0.1	0.333	0.333	0.333	1.113	0.692
0.1	0.457	0.170	0.170	1.116	<b>0.712</b>
0.01	0.333	0.333	0.333	1.243	0.685
0.01	0.455	0.176	0.369	1.254	0.706
0.001	0.333	0.333	0.333	1.435	0.676
0.001	0.405	0.192	0.403	1.458	0.686

The best MAP found for each combination is highlighted in bold



**Fig. 11** Effectiveness of combining from 1 up to 8 descriptors using  $\alpha$ -normalization and weighting by Max- $\tau$  for the average and keyframe descriptors

Table 3b shows the combination C44+K11. In this case the correlation between MAP and  $\tau_{\alpha,\delta}$  is stronger (particularly for  $\alpha = 0.01$  and 0.001). Thus, selecting a set of weights with higher  $\tau_{\alpha,\delta}$  is close related to an improvement in MAP, with the

**Table 3** Effectiveness (MAP) for a combination of average descriptors for different weights and for different  $\alpha$ -normalization (segments of one second length,  $W = 1$ )

Combination of average descriptors EH10 and C14										
$\delta$	$w_1$ (EH10)	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
	$w_2$ (C14)	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
MAP	$\alpha = 1$	0.525	0.547	0.569	0.599	0.636	0.672	0.705	<b>0.729</b>	0.722
	$\alpha = 0.1$	0.534	0.561	0.593	0.633	0.670	0.701	0.725	<b>0.729</b>	0.713
	$\alpha = 0.01$	0.532	0.558	0.588	0.626	0.664	0.695	0.723	<b>0.730</b>	0.715
	$\alpha = 0.001$	0.530	0.556	0.582	0.618	0.656	0.690	0.719	<b>0.730</b>	0.717
$\tau_{\alpha,\delta}$	$\alpha = 1$	<b>0.981</b>	0.979	0.975	0.954	0.953	0.960	0.965	0.975	0.978
	$\alpha = 0.1$	1.021	1.038	1.052	1.061	<b>1.066</b>	1.065	1.057	1.044	1.025
	$\alpha = 0.01$	1.045	1.083	1.113	1.135	1.148	<b>1.149</b>	1.137	1.109	1.064
	$\alpha = 0.001$	1.074	1.135	1.184	1.219	1.238	<b>1.240</b>	1.221	1.180	1.108

Combination of average descriptors C44 and K11										
$\delta$	$w_1$ (C44)	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
	$w_2$ (K11)	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
MAP	$\alpha = 1$	0.552	0.565	<b>0.566</b>	0.559	0.550	0.542	0.534	0.525	0.516
	$\alpha = 0.1$	0.539	0.554	0.564	<b>0.567</b>	0.565	0.556	0.547	0.536	0.524
	$\alpha = 0.01$	0.540	0.555	0.564	<b>0.566</b>	0.563	0.555	0.545	0.535	0.522
	$\alpha = 0.001$	0.544	0.559	0.566	<b>0.566</b>	0.559	0.551	0.541	0.532	0.520
$\tau_{\alpha,\delta}$	$\alpha = 1$	<b>1.000</b>	0.997	0.997	0.999	0.997	0.999	0.998	0.999	0.999
	$\alpha = 0.1$	1.014	1.027	1.037	1.044	<b>1.047</b>	1.046	1.040	1.031	1.017
	$\alpha = 0.01$	1.039	1.064	1.077	<b>1.081</b>	1.079	1.072	1.060	1.044	1.024
	$\alpha = 0.001$	1.065	1.102	1.117	<b>1.119</b>	1.109	1.096	1.077	1.057	1.032

For each  $\alpha$ , the highest MAP and  $\tau_{\alpha,\delta}$  are highlighted in bold

advantage that  $\tau_{\alpha, \delta}$  is a statistical value that can be calculated without any ground truth or training database.

#### 4.2.4 Effectiveness vs approximate search

In the next experiment we analyze the performance of the approximate search for average descriptors EH10, C14, and KF11 with segments of one second length and  $W = 3$ . Table 4 summarizes the effectiveness of the exact search, the number of correct detections without false alarms (recall at precision 1), the intrinsic dimensionality for the search space, and the required number of operations to evaluate  $\delta$  (roughly,  $W \times$  descriptor size).

The approximate search (Algorithm 3) will achieve the same effectiveness as the exact search when most of the correct answers are located between the  $T$  smallest values of  $LB_{\mathcal{P}}$ . Figure 12 shows the amount of query segments whose  $LB_{\mathcal{P}}$  for the correct answer is located between the  $T$  smallest values of  $LB_{\mathcal{P}}$  (note the logarithmic scale in the value of  $T$ ). We have compared the performance for sets of 5, 10, 20 and 40 pivots for the three descriptors. Because the pivot selection algorithm has a random component, we selected many sets of pivots and we plot the average, the best and the worst obtained values. The Fig. 12 first shows that the approximation improves with the number of pivots. It also shows that the improvements are decreasing: the improvement from 5 to 10 pivots is higher than the improvement from 10 to 20, and the latter is higher than 20 to 40. The approximation performance of the pivots also depends on the descriptor: the most difficult descriptor to approximate is EH10, while the approximation for C14 and K11 achieves an almost similar performance.

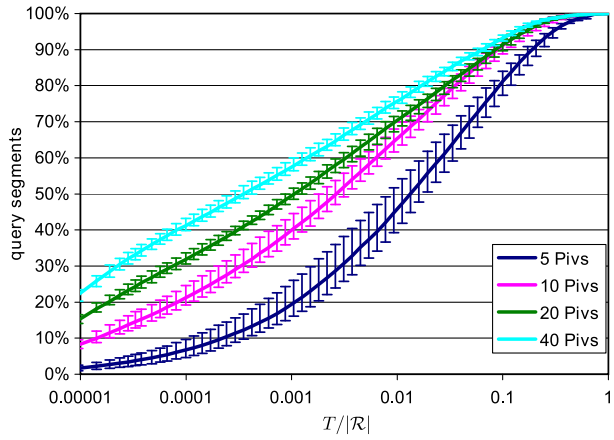
For example in EH10, an approximate search that calculates  $LB_{\mathcal{P}}$  with 10 pivots, selects just the 1% of the smallest  $LB_{\mathcal{P}}$ , and evaluates  $\delta$  just for them (discarding the other 99% of objects), achieves a similar performance that the exact search for at least 64% of the queries. In fact, the MAP for the approximate search is 0.553, and the number of correct answers are  $N1 = 51.4\%$  and  $N10 = 60.6\%$ . The search time is reduced by 95% compared with Algorithm 2, the search effectiveness is reduced by a 32%, and the detection performance decreases 7% (recall 0.871 with precision 1). For  $T = 0.1|\mathcal{R}|$  the MAP is 0.754, with  $N1 = 70.0\%$  and  $N10 = 82.9\%$ , the search time is reduced by about a 79%, the search effectiveness is reduced by about a 7%, and the detection performance decreases 3.5% (recall 0.903 with precision 1).

The search time of the approximate search depends on  $T$ ,  $|\mathcal{P}|$  and  $\delta$ . Figure 13 summarizes the total search time for the approximate search. The graphs show that the search time grows almost linearly with  $T$  independently of the descriptor.

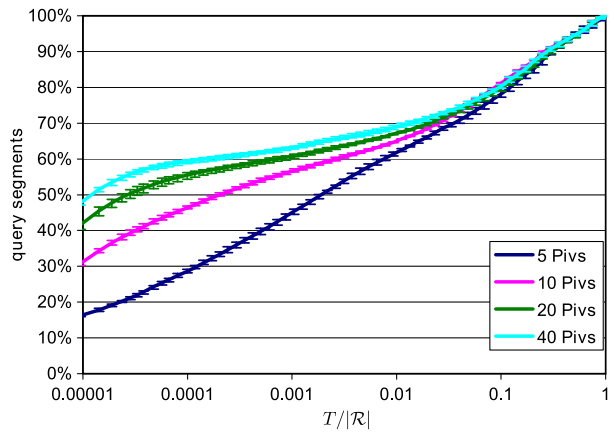
**Table 4** Effectiveness of average global descriptors (segments of one second length,  $W = 3$ )

	EH10	C14	K11
MAP	0.810	0.611	0.670
N1	0.752	0.557	0.610
N10	0.894	0.685	0.765
recall	0.935	0.710	0.742
$\rho$	10.80	9.66	4.50
#opers.	480	576	297

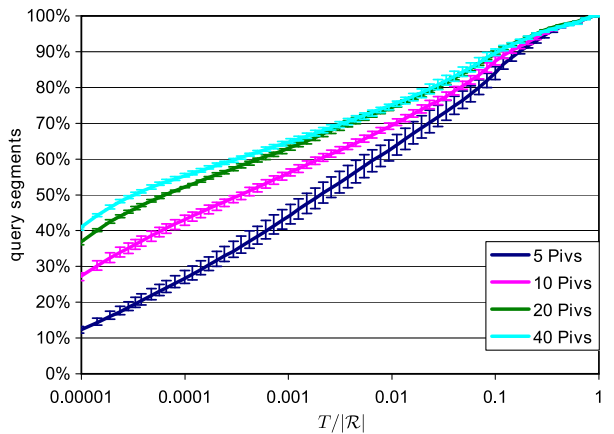
**Fig. 12** Proportion of query segments whose correct answer is between the  $T$  smallest values of  $LB_{\mathcal{P}}$ , for each  $|\mathcal{P}| \in \{5, 10, 20, 40\}$ . The best, worst and average value is shown. Logarithmic scale



(a) EH10 (W=3).

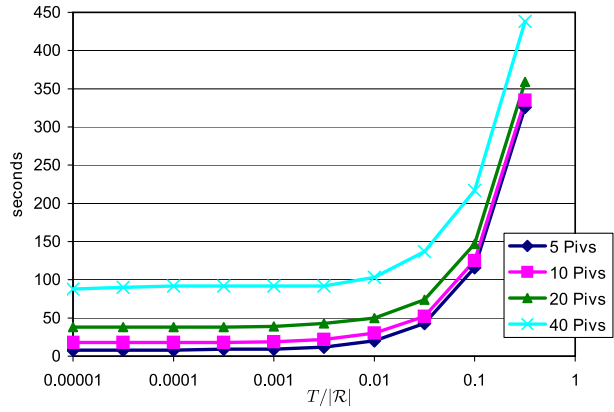


(b) C14 (W=3).

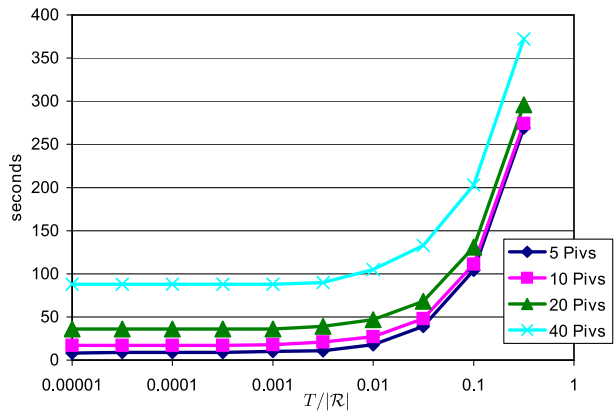


(c) K11 (W=3).

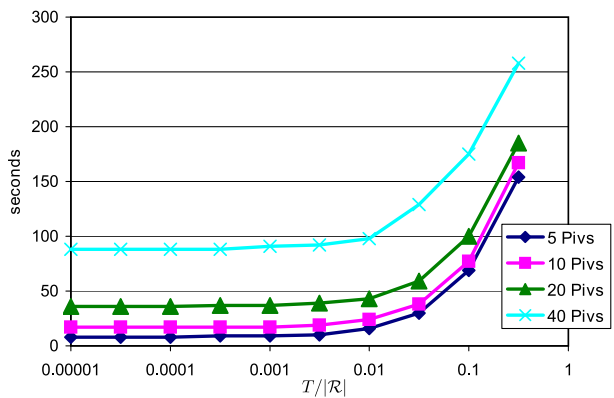
**Fig. 13** Computational time for  $|\mathcal{Q}|$  approximate searches for  $T$  and  $|\mathcal{P}| \in \{5, 10, 20, 40\}$ . Logarithmic scale



(a) EH10 ( $W=3$ ).



(b) C14 ( $W=3$ ).



(c) K11 ( $W=3$ ).

In summary, the effectiveness of the approximate search grows almost exponentially in  $T$ , while its search time grows linearly. This behavior shows that the approximate search is effective and efficient.

### 4.3 TRECVID CCD 2010 evaluation

As a part of TRECVID since 2008, the content-based copy detection task (CCD) evaluates CBVCD systems with large video collections and uniform scoring procedures. Given a collection of reference videos and a collection of query videos, the goal of the CCD task is to determine if a query video contains a sequence that originates from a reference video (with possible transformations), and to localize the original sequences and the duplicated.

TRECVID 2010 considered a reference dataset of around 11K files, with individual lengths between 10 s and 30 min, and a total duration of more than 400 h. All these files were collected from video-sharing websites in an aim to bring video copy detection close to a real-world application scenario. A variety of formats and kinds of videos were present, as for example movie excerpts, animation videos, mobile phone videos, user generated videos, slideshows, etc.

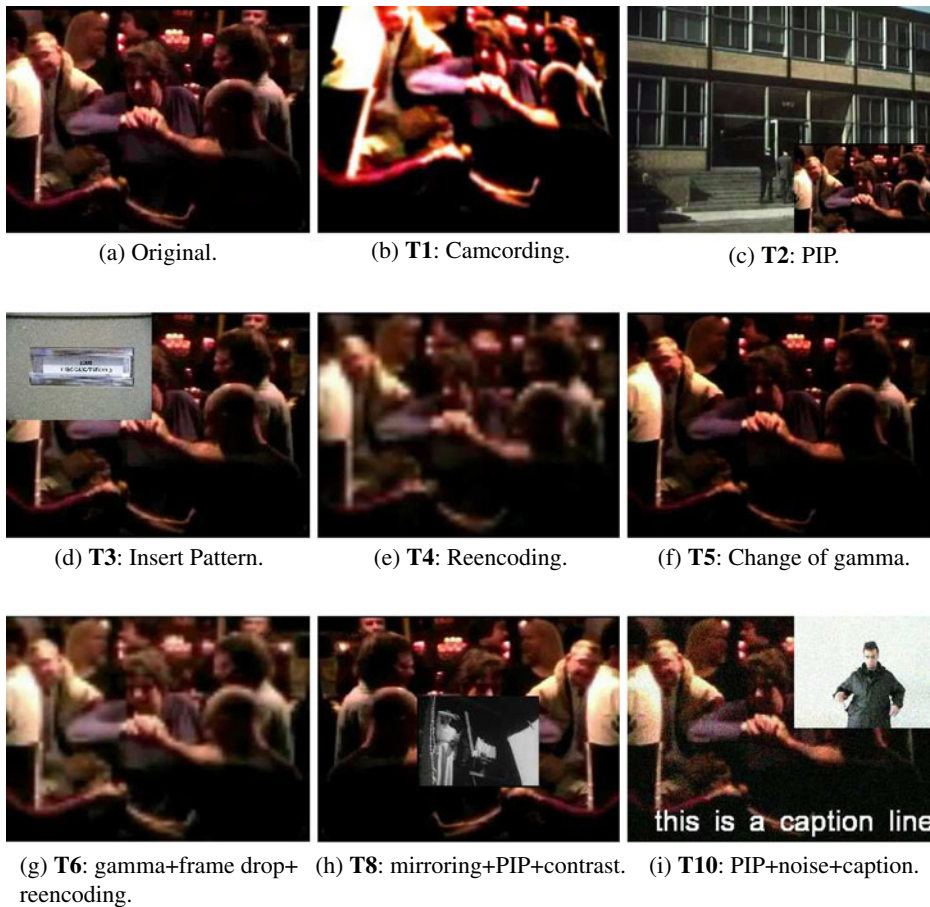
The query dataset was created by first selecting 201 base videos each one with a length between 3s and 3m: 67 videos were an excerpt from a video in the reference dataset, 67 were an excerpt from a video not in the reference dataset, and 67 were an excerpt from a video in the reference dataset embedded into a longer excerpt from a video not in the reference dataset. Then, 56 transformations were applied to each base video (a combination between 8 visual transformations and 7 audio transformations) creating around 11K query videos where two thirds of them were copies. The eight visual transformations evaluated in TRECVID 2010 were: **T1**: simulated camcording; **T2**: picture-in-picture (PIP) original video in foreground; **T3**: insertion of pattern; **T4**: strong reencoding; **T5**: change of gamma; **T6**: three transformations between blur, change of gamma, frame dropping, contrast, reencoding, ratio, and white noise; **T8**: three transformations between crop, shift, contrast, caption, mirroring, insertion of pattern, and PIP original video in background; **T10**: random combination of three previous transformations; **T7** and **T9** were not evaluated. Figure 14 shows these transformations applied to a base video.

All participant systems should use visual and audio information for copy detection. However, P-VCD combined only visual descriptors, discarding all audio information.

The evaluation is based on three measures: NDCR, which measures the effectiveness of the detection weighting the probability of missing a detection and the probability to falsely indicate that there is a copy for a query video ( $NDCR = P_{MISS} + \beta \cdot P_{FA}$ , the closer to zero the better the effectiveness); F1, which measures the accuracy in localization after a copy has been correctly detected (the closer to 1 the better the accuracy); and Mean Processing Time, which measures the efficiency for processing queries. NDCR was evaluated for two profiles: Balanced ( $\beta = 200$ ) and No False Alarms (NoFA,  $\beta = 200,000$ ). TRECVID calculates these measures separately for each transformation, and for comparison purposes we include the average result for all transformations.

#### 4.3.1 TRECVID 2010 results

Twenty-two teams participated in the evaluation. Each team submitted 4 Runs, which resulted in 37 submissions for NoFA profile (with 14 visual-only Runs), and 41 submissions for Balanced profile (with 15 visual-only Runs). We stated that a Run is



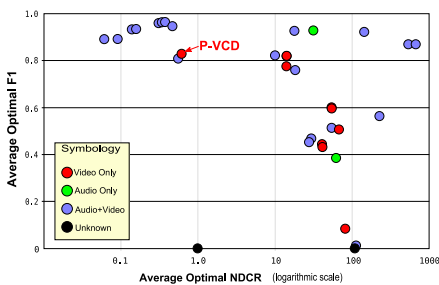
**Fig. 14** Example of transformations in TRECVID 2010. **a** Frame from a reference video, **b–i** same frame from each query video for the eight visual transformations

visual-only when its results were independent of changes in audio (i.e. when NDCR, F1 and processing time did not change for the 7 audio transformations in a same visual transformation).

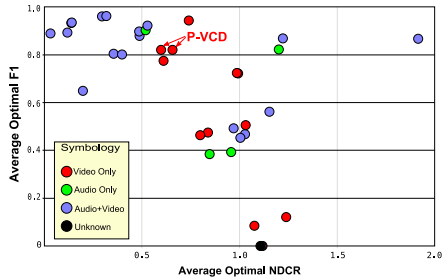
Figure 15 shows the results for NoFA profile and Balanced profile. In the NoFA profile, we tested a combination of EH10 descriptors and a G33 (gray histogram of 20 bins for  $3 \times 3$  zones). It achieved a better NDCR than the median for every transformation, and considering just visual-only Runs it achieved the best results for average NDCR and F1. For the Balanced profile, we tested two combinations: (1) EH10 and G33 descriptors, which achieved the best results for NDCR considering just visual-only Runs, and (2) EH10 and C22 (RGB histogram of 16 bins for each channel for  $2 \times 2$  zones) which improved NDCR and F1 for some transformations but the average results were slightly worse than EH10 and G33. For all Runs, the parameters for Algorithm 3 were  $|\mathcal{P}| = 9$  and  $T = 0.001|\mathcal{R}|$ . In (4), we fix  $W = 3$  thus  $\delta$  function needs more than 1,000 operations to be evaluated, but  $LB_{\mathcal{P}}$  estimated it with just 9 operations. Reference and query videos produced  $|\mathcal{R}| = 3,967,815$

	Average	Rank	Visual-Only
NoFA Run – NDCR	0.611	10 <sup>th</sup> of 37	1 <sup>st</sup> of 14
NoFA Run – F1	0.828	14 <sup>th</sup> of 37	1 <sup>st</sup> of 14
NoFA Run – Proc. Time	128 s.	23 <sup>th</sup> of 37	11 <sup>th</sup> of 14
Balanced Run 1 – NDCR	0.597	14 <sup>th</sup> of 41	1 <sup>st</sup> of 15
Balanced Run 1 – F1	0.820	17 <sup>th</sup> of 41	3 <sup>rd</sup> of 15
Balanced Run 1 – Proc. Time	128 s.	27 <sup>th</sup> of 41	11 <sup>th</sup> of 15
Balanced Run 2 – NDCR	0.658	16 <sup>th</sup> of 41	3 <sup>rd</sup> of 15
Balanced Run 2 – F1	0.820	16 <sup>th</sup> of 41	2 <sup>nd</sup> of 15
Balanced Run 2 – Proc. Time	132 s.	28 <sup>th</sup> of 41	12 <sup>th</sup> of 15

(a) Evaluation for submitted Runs. Average values for the 56 transformations at the optimal decision threshold.



(b) Submissions for NoFA profile.



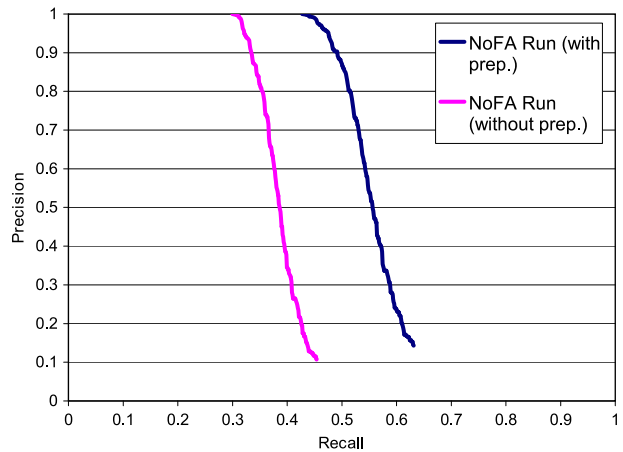
(c) Submissions for Balanced profile.

**Fig. 15** P-VCD’s results for TRECVID 2010 CCD task compared with all the other participants. A system with optimal performance would have NDCR = 0 and F1 = 1

reference segments and  $|\mathcal{Q}| = 990,246$  query segments, thus actual  $\delta$  was evaluated just 3,967 times for each query segment. More details about the implementation and results by transformation are reviewed in [2].

The preprocessing tasks has a big impact in the system performance. It tries to detect and revert the transformations of camcording (T1), PIP (T2), and vertical mirroring (T8) by creating new query videos. Starting with the original 1,608 queries, the task added 3,770 new queries to the system. Thus, the search times increased in more than three times with the benefit that the descriptors are invariant to those three transformations. Figure 16 shows a precision/recall graph for the NoFA submission together with the results if the preprocessing task is not present (i.e. discarding detection from query videos with some of those transformations because they are undetectable). The recall at precision 1 decreases from 0.43 to 0.30, and the recall at precision 0.5 decreases from 0.56 to 0.39. On the other hand, the search times are reduced in almost a 70%, thus there is more room for improving the accuracy of the approximate search.

**Fig. 16** Precision/recall graph for NoFA run and for NoFA discarding the detection from queries with camcording, PIP and mirroring

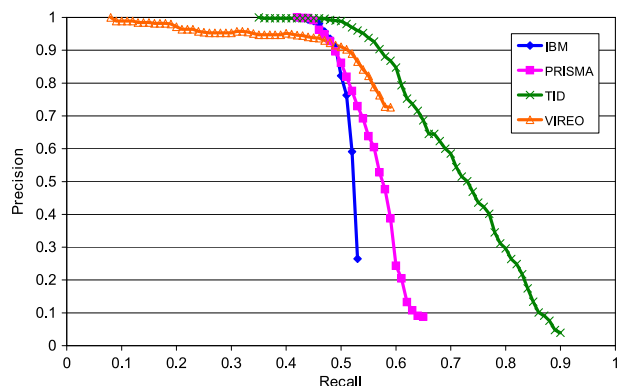


#### 4.3.2 Analysis of P-VCD results

After a careful analysis of the results, it becomes clear that the major strength of P-VCD is its performance for the No False Alarms profile. The NDCR sets a higher cost to false alarms than to miss detections (even in Balanced profile). Before returning the first false alarm, P-VCD detected many query videos from all the transformations, then its average NDCR is low.

Figure 17 shows a precision/recall graph with the four best visual-only submissions (teams IBM [23], TID [31], VIREO [24], and PRISMA [2]). P-VCD achieves the highest recall at precision 1, and then as precision decreases, the other systems (which are based on local descriptors) improves their recall and outperform P-VCD. A reason for this behavior is that non-copy query videos are usually easy to discard for global descriptors but can be difficult to discard for local descriptors. For example, two unrelated videos that share captions or logos can trigger a false detection under local descriptors, however for global descriptor this is unlikely because captions and logos only change small zones. Then, the amount of correct detections up to the first false alarm can be higher for global descriptors than for local descriptors, however as precision decreases the recall for local descriptors

**Fig. 17** Precision/recall graph for four best visual-only submissions of TRECVID 2010



can outperform global descriptors because some complex transformations may be undetectable under global descriptors.

## 5 Conclusions and final comments

In this work we have reviewed in detail P-VCD and its experimental evaluation. The results show that this approach enables global descriptors to achieve competitive results and even outperform systems based on combination of local descriptors and audio information.

A general property that a global descriptor should comply with is the division in independent zones. For example, a gray histogram of 180 bins for the whole frame achieves poor results for detecting copies, however if we divide the frames in  $3 \times 3$  zones and calculate a gray histogram of 20 bins for each zone, the descriptor size is kept but the detection performance improves greatly.<sup>2</sup> This is because some usual transformations affect specific zones of the video (like insertion of captions and logos) leaving unchanged the others. In the case of transformations that can affect the whole zoning (like mirroring or PIP) a preprocessing should be considered to create one or more query videos with the transformations reverted.

Another property for global descriptor is the use of the average descriptor for a segment instead of the descriptor just for the keyframe. Moreover, the performance can be improved even more by using a temporal distance between segments. The temporal relationship between frames has already been used in other systems for different kinds of descriptors, but here we show an approach that is general enough to use it with most of the global descriptors.

A remarkable reason for the satisfactory performance of this system is the use of the metric space approach. Most of the systems that use global descriptors calculate as few dimensions as possible (usually between 5 and 20 dimensions) to avoid the well known curse of dimensionality. We extract global descriptors with high dimensionality (for example, more than 200 dimensions) that achieve high effectiveness, because MAMs do not index vectors using their dimensions but using the distances between them. Even though the metric spaces also present the curse of dimensionality issue, it is more associated with the distribution of distances between objects rather than the nominal dimension of the vectors (as is measured by the intrinsic dimensionality indicator).

It is well known that local descriptors should be used for complex transformations [19]. However, global descriptors can achieve competitive results and even outperform many systems that use local descriptors and/or combine visual and audio information. For this, a combination of global descriptors, a smart weighting algorithm and a good approximate search are required.

An interesting property of this approach is that the  $\delta$  function in (4) combines descriptors at the similarity search level, thus enabling the fusion of descriptors at an earlier stage rather than at the final decision level. The combination of global descriptors, local descriptors, and audio information can be made by implementing

---

<sup>2</sup>In fact, under the same conditions from Fig. 8 with segments of one second length, the detection performance for this example increases from recall 0.484 (15 detected copies) to 0.710 (22 detected copies) for precision 1.

a feature extraction method for each source of information that complies with the restrictions stated in Section 3.3. Thus, this approach has a potential of achieving even higher effectiveness due to its seamless ability of combining descriptors from different sources.

## References

1. Anguera X, Obrador P, Oliver N (2009) Multimodal video copy detection applied to social media. In: Proc. of the 1st SIGMM workshop on social media (WSM'09). ACM, pp 57–64
2. Barrios J, Bustos B (2010) Content-based video copy detection: PRISMA at TRECVID 2010. In: TRECVID. NIST
3. Barrios J, Bustos B (2011) P-VCD: a pivot-based approach for content-based video copy detection. In: Proc. of the IEEE int. conf. on multimedia and expo (ICME'11). IEEE, pp 1–6
4. Batko M, Kohoutkova P, Novak D (2009) A Cophir image collection under the microscope. In: Proc. of the intl. workshop on similarity search and applications (SISAP'09). IEEE, pp 47–54
5. Bay H, Ess A, Tuytelaars T, Gool LV (2008) Speeded-up robust features (SURF). *Comput Vis Image Underst* 110(3):346–359
6. Brin S (1995) Near neighbor search in large metric spaces. In: Proc. of the int. conf. on very large databases (VLDB'95). Morgan Kaufman, pp 574–584
7. Bustos B, Skopal T (2006) Dynamic similarity search in multi-metric spaces. In: Proc. of the int. workshop on multimedia information retrieval (MIR'06). ACM, pp 137–146
8. Bustos B, Pedreira O, Brisaboa N (2008) A dynamic pivot selection technique for similarity search. In: Proc. of the int. workshop on similarity search and applications (SISAP'08). IEEE, pp 105–112
9. Chávez E, Navarro G, Baeza-Yates R, Marroquín JL (2001) Searching in metric spaces. *ACM Comput Surv* 33(3):273–321
10. Ciaccia P, Patella M, Zezula P (1997) M-tree: an efficient access method for similarity search in metric spaces. In: Proc. of the int. conf. on very large databases (VLDB'97). Morgan Kaufman, pp 426–435
11. Deselaers T, Weyand T, Ney H (2007) Image retrieval and annotation using maximum entropy. In: CLEF Workshop 2006. LNCS, vol 4730. Springer, pp 725–734
12. Douze M, Gaidon A, Jegou H, Marszalek M, Schmid C (2008) INRIA LEAR's video copy detection system. In: TRECVID. NIST
13. Gupta V, Boulianne G, Cardinal P (2010) CRIM's content-based audio copy detection system for TRECVID 2009. In: Proc. of the int. workshop on content-based multimedia indexing (CBMI'10). IEEE
14. Hampapur A, Bolle R (2001) Comparison of distance measures for video copy detection. In: Proc. of the IEEE int. conf. on multimedia and expo (ICME'01). IEEE, pp 737–740
15. Joly A, Buisson O, Frélicot C (2007) Content-based copy retrieval using distortion-based probabilistic similarity search. *IEEE Trans Multimedia* 9(2):293–306
16. Kim C, Vasudev B (2005) Spatiotemporal sequence matching for efficient video copy detection. *IEEE Trans Circuits Syst Video Technol* 15(1):127–132
17. Law-To J, Joly A, Boujemaa N (2007) MUSCLE-VCD-2007: a live benchmark for video copy detection. <http://www-rocq.inria.fr/imedia/civr-bench/>
18. Law-To J, Buisson O, Gouet-Brunet V, Boujemaa N (2006) Robust voting algorithm based on labels of behavior for video copy detection. In: Proc. of the int. conf. on multimedia (ACMMM'06), pp 835–844. ACM
19. Law-To J, Chen L, Joly A, Laptev I, Buisson O, Gouet-Brunet V, Boujemaa N, Stentiford F (2007) Video copy detection: a comparative study. In: Proc. of the int. conf. on image and video retrieval (CIVR'07). ACM, pp 371–378
20. Lew M, Sebe N, Djeraba C, Jain R (2006) Content-based multimedia information retrieval: state of the art and challenges. *ACM Transactions on Multimedia Computing, Communications and Applications* 2(1):1–19
21. Lowe D (2004) Distinctive image features from scale-invariant keypoints. *Int J Comput Vis* 60(2):91–110
22. Manjunath BS, Ohm JR, Vasudevan VV, Yamada A (2001) Color and texture descriptors. *IEEE Trans Circuits Syst Video Technol* 11(6):703–715

23. Natsev A, Smith JR, Hill M, Hua G, Huangy B, Merlery M, Xie L, Ouyangz H, Zhoux M (2010) IBM research TRECVID-2010 video copy detection and multimedia event detection system. In: TRECVID. NIST
24. Ngo CW, Zhu SA, Tan HK, Zhao WL, Wei XY (2010) VIREO at TRECVID 2010: semantic indexing, known-item search, and content-based copy detection. In: TRECVID. NIST
25. Poullot S, Buisson O, Crucianu M (2007) Z-grid-based probabilistic retrieval for scaling up content-based copy detection. In: Proc. of the int. conf. on image and video retrieval (CIVR'07). ACM, pp 348–355
26. Poullot S, Crucianu M, Buisson O (2008) Scalable mining of large video databases using copy detection. In: Proc. of the int. conf. on multimedia (ACMMM'08). ACM, pp 61–70
27. Sivic J, Zisserman A (2003) Video google: a text retrieval approach to object matching in videos. In: Proc. of the IEEE int. conf. on computer vision (ICCV'03), vol 2. IEEE, pp 1470–1477
28. Skopal T (2007) Unified framework for fast exact and approximate search in dissimilarity spaces. *ACM Trans Database Syst* 32(4):29–47
29. Smeaton AF, Over P, Kraaij W (2006) Evaluation campaigns and TRECVID. In: Proc. of the int. workshop on multimedia information retrieval (MIR'06). ACM, pp 321–330
30. Smeulders AWM, Worring M, Santini S, Gupta A, Jain R (2000) Content-based image retrieval at the end of the early years. *IEEE Trans Pattern Anal Mach Intell* 22(12):1349–1380
31. Younessian E, Anguera X, Adamek T, Oliver N, Marimon D (2010) Telefonica research at TRECVID 2010 content-based copy detection. In: TRECVID. NIST
32. Zezula P, Amato G, Dohnal V, Batko M (2005) *Similarity search: the metric space approach (advances in database systems)*. Springer



**Juan Manuel Barrios** is PhD candidate at the Department of Computer Science, University of Chile. Currently, he is research assistant at the PRISMA Research Group. His research interests include video copy detection, pattern recognition, and multimedia retrieval. He obtained a M.Sc. degree in Computer Science from the University of Chile in 2006.



**Benjamin Bustos** is an Assistant Professor at the Department of Computer Science, University of Chile. He is head of the PRISMA Research Group. He leads research projects in the domains of multimedia retrieval, multimedia databases, video copy detection, sketch-based image retrieval, and image processing. His research interests include similarity search, multimedia information retrieval, 3D object retrieval, (non)- metric indexing, and pattern recognition. Benjamin Bustos obtained doctoral degree in natural sciences from the University of Konstanz, Germany, in 2006.